

EXPRESS MAIL MAILING LABEL NUMBER **EL4216463465US**

DATE OF DEPOSIT **9/3/99**

I HEREBY CERTIFY THAT THIS PAPER OR FEE IS BEING DEPOSITED WITH THE UNITED STATES POSTAL SERVICE "EXPRESS MAIL POST OFFICE TO ADDRESSEE" SERVICE UNDER 37 CFR 1.10 ON THE DATE INDICATED ABOVE AND IS ADDRESSED TO THE COMMISSIONER OF PATENTS AND TRADEMARKS, WASHINGTON, D.C. 20031

Eric Link
(TYPED OR PRINTED NAME OF PERSON MAILING PAPER OR FEE)

Eric Link
(SIGNATURE OF PERSON MAILING PAPER OR FEE)

Case Docket No. 98AG07053137

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

Transmitted herewith for filing is the patent application of:

Inventors: **D. PAU, R. SANNINO, A. CAPASSO, P. FRAGNETO**

For: **METHOD AND SCALABLE ARCHITECTURE FOR PARALLEL CALCULATION OF THE DCT OF BLOCKS OF PIXELS OF DIFFERENT SIZES AND COMPRESSION THROUGH FRACTAL CODING**

Enclosed are:

- (X) Patent Application: 46 pages, 4 claims.
- (X) 24 Sheets of drawings.
- (X) A certified copy of European Priority Application Number 98930522.3
- (X) Citation Under 37 CFR 1.97 and PTO-1449.
- (X) Preliminary Amendment.
- (X) Submission of Proposed Drawing Modification.

The Declaration and Filing Fee are **NOT ENCLOSED**.

Name, Address and Citizenship of Inventor(s) is as follows:

Danilo PAU
Via Dante, 131
20099 Sesto San Giovanni, Italy
Citizen of Italy

Andrea CAPASSO
Via De Giosa, 79
70100 Bari, Italy
Citizen of Italy

Roberto SANNINO
Via Duca d'Aosta, 4
24058 Romano Lombardo, Italy
Citizen of Italy

Pasqualina FRAGNETO
Via Roma, 69
82038 Vitulano, Italy
Citizen of Italy

PLEASE ADDRESS ALL CORRESPONDENCE TO ATTORNEY OF RECORD

CHRISTOPHER F. REGAN
Reg. No. 34,906
Allen, Dyer, Doppelt, Milbrath
& Gilchrist, P.A.
255 S. Orange Avenue, Suite 1401
P.O. Box 3791
Orlando, Florida 32802-3791
Phone: (407) 841-2330

September 3, 1999
Date

CHRISTOPHER F. REGAN
Reg. No. 34,906

Allen, Dyer, Doppelt, Milbrath & Gilchrist, P.A.
Attorneys-At-Law, 255 S. Orange Avenue, Suite 1401, P.O. Box 3791, Orlando, Florida 32802-3791

In re Patent Application of
PAU ET AL.
Serial No. **Not Yet Assigned**
Filed: **Herewith**

5. A method of calculating the discrete cosine transform (DCT) of blocks of pixels of an image, comprising the steps of:

defining first subdivision blocks as range blocks, having a fractional and scalable size $N/2^i * N/2^i$, where i is an integer;

defining second subdivision blocks of $N*N$ pixels as domain blocks, shiftable by intervals of $N/2^i$ pixels; and

calculating, in parallel, the DCT of 2^i range blocks of a domain block of $N*N$ pixels of the image.

6. A method according to Claim 5, wherein the step of calculating comprises the steps of:

a) ordering the pixels in the range blocks of a certain dimension by rearranging input pixels in 2^i vectors of 2^i components;

b) calculating, in parallel, 2^i monodimensional DCTs by processing the vectors defined in the step a);

c) arranging output sequences of the monodimensional DCTs relative to the 2^i vectors;

d) completing the calculation in parallel of 2^i bidimensional DCTs by processing output sequences of monodimensional DCTs produced in step c); and

e) arranging output sequences of bidimensional DCTs generated in step d) in 2^i vectors of bidimensional DCT coefficients.

7. A method according to Claim 6, wherein the step of calculating 2^i monodimensional DCTs in parallel in step b)

In re Patent Application of
PAU ET AL.
Serial No. **Not Yet Assigned**
Filed: **Herewith**

and the step of completing the parallel calculation of 2^i bidimensional DCTs of step d) are performed by subdividing the sequences resulting from step a) and from step c), respectively, in groups of scalar elements, calculating the sums and differences thereof by way of adders and subtractors and by reiterately multiplying the sum and difference results by respective coefficients until completing the calculation of the relative DCT coefficients, respectively monodimensional and bidimensional.

8. A method of compressing data of an image to be stored or transmitted, comprising the steps of:

defining first subdivision blocks as range blocks, having a fractional and scalable size $N/2^i * N/2^i$, where i is an integer;

defining second subdivision blocks of $N*N$ pixels as domain blocks, shiftable by intervals of $N/2^i$ pixels;

calculating, in parallel, the DCT of 2^i range blocks and of a relative domain block;

classifying the transformed range blocks according to their relative complexity represented by a sum of values of three AC coefficients;

applying a fractal transform in the DCT domain to data of the range blocks whose complexity classification exceeds a pre-defined threshold and only storing a DC coefficient of the range blocks with a complexity lower than the threshold, while identifying a relative domain block to which the range block in a transformation belongs that produces a best fractal approximation of the range block;

In re Patent Application of
PAU ET AL.
Serial No. **Not Yet Assigned**
Filed: **Herewith**

calculating a difference between each range block and its fractal approximation;

quantizing the difference in the DCT domain by using a quantization table preestablished in consideration of human sight characteristics;

coding the quantized difference by a process based on probabilities of quantization coefficients; and

storing or transmitting code of each range block compressed in the DCT domain and the DC coefficient of each uncompressed range block.

9. An apparatus for calculating the discrete cosine transform (DCT) of blocks of pixels of an image, the apparatus comprising:

means for defining first subdivision blocks as range blocks, having a fractional and scalable size $N/2^i * N/2^i$, where i is an integer;

means for defining second subdivision blocks of $N*N$ pixels as domain blocks, shiftable by intervals of $N/2^i$ pixels; and

means for calculating, in parallel, the DCT of 2^i range blocks of a domain block of $N*N$ pixels of the image.

10. An apparatus according to Claim 9, wherein the means for calculating comprises:

means for ordering the pixels in the range blocks of a certain dimension by rearranging input pixels in 2^i vectors of 2^i components;

In re Patent Application of
PAU ET AL.
Serial No. **Not Yet Assigned**
Filed: **Herewith**

means for calculating, in parallel, 2^i monodimensional DCTs by processing the vectors defined by the means for calculating;

means for arranging output sequences of the monodimensional DCTs relative to the 2^i vectors;

means for completing the calculation in parallel of 2^i bidimensional DCTs by processing output sequences of monodimensional DCTs produced by the means for arranging output sequences of the monodimensional DCTs; and

means for arranging output sequences of bidimensional DCTs, generated by the means for completing the calculation, in 2^i vectors of bidimensional DCT coefficients.

11. An apparatus according to Claim 10, wherein the means for calculating 2^i monodimensional DCTs in parallel in and the means for completing the parallel calculation of 2^i bidimensional DCTs are for subdividing the sequences resulting from the means for ordering and the means for arranging output sequences of the monodimensional DCTs, respectively, in groups of scalar elements, calculating the sums and differences thereof by way of adders and subtractors and by reiterately multiplying the sum and difference results by respective coefficients until completing the calculation of the relative DCT coefficients, respectively monodimensional and bidimensional.

12. An apparatus for compressing data of an image to be stored or transmitted, comprising:

In re Patent Application of
PAU ET AL.
Serial No. **Not Yet Assigned**
Filed: **Herewith**

means for defining first subdivision blocks as range blocks, having a fractional and scalable size $N/2^i * N/2^i$, where i is an integer;

means for defining second subdivision blocks of $N * N$ pixels as domain blocks, shiftable by intervals of $N/2^i$ pixels;

means for calculating, in parallel, the DCT of 2^i range blocks and of a relative domain block;

means for classifying the transformed range blocks according to their relative complexity represented by a sum of values of three AC coefficients;

means for applying a fractal transform in the DCT domain to data of the range blocks whose complexity classification exceeds a pre-defined threshold and only storing a DC coefficient of the range blocks with a complexity lower than the threshold, while identifying a relative domain block to which the range block in a transformation belongs that produces a best fractal approximation of the range block;

means for calculating a difference between each range block and its fractal approximation;

means for quantizing the difference in the DCT domain by using a quantization table preestablished in consideration of human sight characteristics;

means for coding the quantized difference by a process based on probabilities of quantization coefficients;
and

means for storing or transmitting code of each range block compressed in the DCT domain and the DC coefficient of each uncompressed range block.

In re Patent Application of
PAU ET AL.

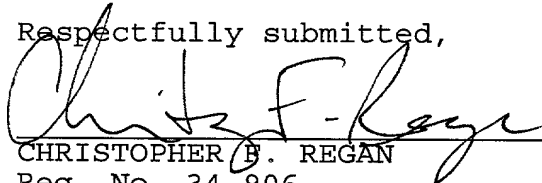
Serial No. Not Yet Assigned

Filed: Herewith

REMARKS

It is believed that all of the claims are patentable over the prior art. Accordingly, after the Examiner completes a thorough examination and finds the claims patentable, a Notice of Allowance is respectfully requested in due course. Should the Examiner determine any minor informalities that need to be addressed, he is encouraged to contact the undersigned attorney at the telephone number below.

Respectfully submitted,



CHRISTOPHER B. REGAN

Reg. No. 34,906

Allen, Dyer, Doppelt, Milbrath
& Gilchrist, P.A.

255 S. Orange Avenue, Suite 1401

Post Office Box 3791

Orlando, Florida 32802

407-841-2330

407-841-2343 fax

Attorney for Applicant

**METHOD AND SCALABLE ARCHITECTURE FOR PARALLEL
CALCULATION OF THE DCT OF BLOCKS OF PIXELS OF DIFFERENT
SIZES AND COMPRESSION THROUGH FRACTAL CODING**

5

Field of the Invention

The invention relates in general to digital processing systems for recording and/or transmitting pictures, and more in particular to systems for compressing and coding pictures by calculating the discrete cosine transform (DCT) of blocks of pixels of a picture. The invention is particularly useful in video coders according to the MPEG2 standard though it is applicable also to other systems.

15

Background of the Invention

The calculation of the discrete cosine transform (DCT) of a pixel matrix of a picture is a fundamental step in processing picture data. A division by a quantization matrix is performed on the results of the discrete cosine transform for reducing the amplitude of the DCT coefficients, as a precondition to data compression which occurs during a coding phase, according to a certain transfer protocol of video data to be transmitted or stored. Typically, the calculation of the discrete cosine transform is carried out on blocks or matrices of pixels, in which a whole picture is subdivided for processing purposes.

Increasing speed requisites of picture processing systems for storage and/or transmission, imposes the use of hardware architectures to speed up various processing steps among which, primarily, the step of discrete cosine transform calculation by blocks of pixels. Use of hardware processing imposes a pre-definition of few fundamental parameters, namely the dimensions of the blocks of pixels into which a picture is subdivided to meet processing requisites.

Such a pre-definition may represent a heavy constraint that limits the possibility of optimizing the processing system, for example a MPEG2 coder, or its adaptability to different conditions of use in terms of different performance requisites. It is also evident the enormous economic advantage in terms of reduction of costs of an integrated data processing system that may be programmed to calculate in parallel the DCT on several blocks of pixels of size selectable among a certain number of available sizes.

Summary of the Invention

It is evident that a need exists for a method and of a hardware architecture for calculating the discrete cosine transform (DCT) on a plurality of blocks of pixels, in parallel, which provides for the scalability of the size of the blocks of pixels. For example, the calculation of the discrete cosine transform (DCT) either for one block of 8x8 pixels, or four blocks of 4x4 pixels in parallel, or for sixteen blocks of 2x2 pixels in parallel, operating a selection of the block's size.

Scalability of the size of the block of pixels and the possibility of performing the calculation of the discrete cosine transform in parallel on blocks of congruently reduced size compared to a certain maximum block's size, by a hardware structure is also instrumental of the implementation of highly efficient

"hybrid" picture compression schemes and algorithms. For example, by virtue of the scalability of the block size and of the ability to calculate in parallel the DCT on more blocks, it is possible, according to the present invention, to implement a fractal coding applied in the DCT domain rather than in the space domain of picture data, as customary.

Therefore, another important aspect of the invention is a new picture data compressing and coding method that practically is made possible by a hardware structure calculating the DCT on blocks of scaleable size and which includes

subdividing a picture by defining two distinct types of subdivision blocks: a first type, of $N/i * N/i$ dimension called *range* blocks that are not overimposable one on another, and a second type, of $N * N$ dimension, called *domain* blocks, that are transferable by intervals of N/i pixels and overimposable one on another (by transferring on the original picture a window that identifies a *domain* block by an interval equivalent to the horizontal and/or vertical dimension of a *range* block);

calculating the discrete cosine transform (DCT) of the 2^i *range* blocks and of a relative *domain* block in parallel;

classifying the transform *range* blocks according to their relative complexity calculated by summing the three AC coefficients;

applying the fractal transform in the DCT domain to the data of *range* blocks whose complexity exceeds a pre-defined threshold and storing only the DC coefficient of the *range* blocks with a complexity lower than said threshold, identifying a relative *domain* block belonging to the *range* block being transformed that produces the best fractal

appropriate linear transform, for example, rotations, ϕ ,
 overturns, τ , or the like and a domain block DCT, of
 which being defined by $F_D(u,v)$, which at least
 approximately satisfy the following equation:

$$\begin{aligned} F_R(u,v) &= \phi(F_D(u,v)) \\ F_R(u,v) &= \tau(F_D(u,v)) \end{aligned} \quad (1)$$

Having so identified the domain block most similar
 or homologous to the range block that is being
 processed, its parameters $F_D(u,v), \tau, \phi$ are stored. The
 10 difference picture between the range block and its
 fractal position is then calculated:

$$\begin{aligned} D(u,v) &= F_R(u,v) - \phi(F_D(u,v)) \\ D(u,v) &= F_R(u,v) - \tau(F_D(u,v)) \end{aligned} \quad (2)$$

By quantizing the difference picture $D(u,v)$,

$$15 \quad D_Q(u,v) = \text{INTEG}[D(u,v)/Q(u,v)] \quad (3)$$

is obtained, where:

$DQ(u,v)$ is the quantized difference picture in the
 domain of DCT;

$Q(u,v)$ is a quantization table designed by
 20 considering human sight characteristics;

INTEG is a function that approximates its argument
 to the nearest integer;

After quantization, the majority of the $D_Q(u,v)$
 coefficients are null. Therefore, it is easy to design a
 25 coding, for example Huffman coding, based on the
 probabilities of the coefficients. Finally, the code to
 be recorded or transmitted is stored. The compression
 procedure terminates when each range block has been
 coded.

Brief Description of the Drawings

The different aspects and implementations of the scaleable architecture for calculating the discrete cosine transform of the invention as well as of the method of compression and fractal coding, will be more easily understood through the following detailed description of an embodiment of the architecture of the invention and of the different functioning modes according to a selection of the size selection of the blocks of pixels into which the picture is divided by referring to the annexed drawings, wherein:

Figure 1 is a block diagram of a coder effecting hybrid compression based on fractal coding and DCT, according to the present invention;

Figure 2 is a flow chart of the parallel computation of the DCT of sixteen blocks of 2x2-pixel size;

Figure 3 illustrates the architecture for parallel computation of sixteen 2x2 DCTs;

Figure 4 shows the arrangement of the input data for calculating the sixteen 2x2 DCTs;

Figure 5 shows the PROCESS phase of the calculating procedure of sixteen 2x2 DCTs;

Figure 6 illustrates the architecture for parallel computation of four 4x4 DCTs;

Figure 7 shows the arrangement of the input data for calculating the four 4x4 DCTs;

Figure 8 shows the PROCESS phase of the calculating procedure of four 4x4 DCTs;

Figure 9 illustrates the architecture for parallel computation of an 8x8 DCT;

Figure 10 shows the arrangement of the input data for calculating an 8x8 DCT;

Figure 11 shows the PROCESS phase for the calculating procedure of an 8x8 DCT;

Figure 12 shows the scaleable hardware architecture of the invention for calculating an 8x8 DCT or four 4x4 DCTs in parallel or sixteen 2x2 DCTs in parallel;

Figure 13 shows the INPUT structure of the scaleable architecture of the invention;

Figure 14 shows the PROCESS structure for the scaleable architecture of the invention;

Figure 15 shows the functional schemes of the blocks that implement the PROCESS phase in the scaleable architecture of the invention;

Figure 16 is a detailed scheme of the QA block;

Figure 17 is a detailed scheme of the QB block;

Figure 18 is a detailed scheme of the QC block;

Figure 19 is a detailed scheme of the QD block;

Figure 20 is a detailed scheme of the QE block;

Figure 21 is a detailed scheme of the QF block;

Figure 22 is a detailed scheme of the QG block;

Figure 23 illustrates an implementation of the ORDER phase in the scaleable architecture of the invention; and

Figure 24 illustrates an implementation of the OUTPUT phase in the scaleable architecture of the invention.

Detailed Description of the Preferred Embodiments

Though referring in some of the schemes illustrated in the figures to a particularly significant and effective implementation of the architecture of parallel computation of the discrete cosine transform (DCT) on blocks of pixels of scaleable size, which comprises a compression phase for the fractal coding of the picture data, it is understood that the method and architecture of parallel calculation of the discrete cosine transformed (DCT) of a bidimensional matrix of input data by blocks of a scaleable size, provide for an

exceptional freedom in implementing particularly effective compression algorithms by exploiting the scalability and the possibility of a parallel calculation of DCT.

5 The partitioning steps and the calculation of the discrete cosine transform of a bidimensional matrix of input data will be described separately for each size of range block, according to an embodiment of the invention, starting from the smallest block dimension of
10 2x2 for which the DCT calculation is performed in parallel, up to the maximum block dimension of 8x8. This description of an architecture scaleable according to needs by changing the value of the global variable *size*, will follow.

15 The procedure for a parallel DCT computation of the invention may be divided in distinct phases:

 INPUT phase
 PROCESS phase
 ORDER phase
20 OUTPUT phase.

Each phase is hereinbelow described for each case considered.

The DCT operation may be defined as follows.

For an input data matrix $x_{N*N} = [x_{i,j}]_{0 \leq i,j \leq N-1}$, the
25 output matrix $y_{N*N} = [y_{m,n}]_{0 \leq m,n \leq N-1}$, is defined by:

$$y_{m,n} = \frac{2}{N} \varepsilon(m) \varepsilon(n) \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} x_{i,j} \cos\left(\frac{(2i+1)m}{2N} \pi\right) \cos\left(\frac{(2j+1)n}{2N} \pi\right) \quad (4)$$

where:

$$\varepsilon(n) = \begin{cases} \frac{1}{\sqrt{2}} & \text{per } n = 0 \\ 1 & \text{per } 1 \leq n \leq N-1 \end{cases}$$

For convenience, assume that $N=2^i$, where i is an
30 integer and $i \geq 1$. Let's remove $\varepsilon(m)$, $\varepsilon(n)$, and the normalization value $2/N$ from equation (4), in view of

the fact that they may be reintroduced in a successive step. Therefore, from now on, the following simplified version of equation (4) will be used:

$$y_{m,n} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} x_{i,j} \cos\left[\frac{(2i+1)m}{2N}\pi\right] \cos\left[\frac{(2j+1)n}{2N}\pi\right] \quad (5)$$

5

Parallel computation of sixteen 2x2 DCTs

For $N=2$ equation (5) becomes:

$$y_{m,n} = \sum_{i=0}^1 \sum_{j=0}^1 x_{i,j} \cos\left[\frac{(2i+1)m}{4}\pi\right] \cos\left[\frac{(2j+1)n}{4}\pi\right] \quad (6)$$

10 The flow graph for a 2x2 DCT is shown in Fig. 2, in which $A=B=C=1$ and the input and output data are the pixels in the positions $(0,0), (0,1), (1,0), (1,1)$.

Let us now consider how to calculate in parallel sixteen 2x2 DCTs in which an 8x8 block is subdivided. The procedure is divided in many steps, a global view of which is depicted in Fig. 3. This figure highlights the transformations performed on the 2x2 block constituted by the pixels $(0,6), (0,7), (1,6), (1,7)$.

The pixels that constitute the **input block** are ordered in the **INPUT** phase and are processed in the
20 **PROCESS** phase to obtain the coefficients of the sixteen bidimensional DCTs, or briefly 2-D DCTs, on four samples, for example, the 2-D DCT of the block $(0,1)$ constituted by:

$$\{l[0], m[0], n[0], o[0]\} \text{ is } \{a[0], b[0], c[0], d[0]\}$$

25 The coefficients of the 2-D DCT are re-arranged in the **ORDER** phase into eight vectors of eight components. For example the coefficients $\{a[0], b[0], c[0], d[0]\}$ constitute the vector l' . The vectors thus obtained proceed to the **OUTPUT** phase to give the coefficients of
30 the 2x2 DCT, constituting the **output block**.

INPUT phase

The pixels of each block (i,j) , with $0 \leq i \leq 1$ and $0 \leq j \leq 3$, are ordered to the eight-component vectors l, m, n, o in the following manner:

5 the pixels that occupy the position $(0,0)$ in the block constitute the vector l ;

 the pixels that occupy the position $(0,1)$ in the block constitute the vector m ;

10 the pixels that occupy the position $(1,0)$ in the block constitute the vector n ;

 the pixels that occupy the position $(1,1)$ in the block constitute the vector o .

15 Similarly, the pixels of each block (i,j) , with $2 \leq i \leq 3$ and $0 \leq j \leq 3$, are ordered to constitute the eight-component vectors p, q, r, s in the following manner:

 the pixels that occupy the position $(0,0)$ in the block constitute the vector p ;

20 the pixels that occupy the position $(0,1)$ in the block constitute the vector q ;

 the pixels that occupy the position $(1,0)$ in the block constitute the vector r ;

 the pixels that occupy the position $(1,1)$ in the block constitute the vector s .

25 This arrangement is detailed in Fig. 4. It should be noted, for example, that the pixels of the block $(0,3)$ will constitute the third component of the l, m, n, o vectors.

PROCESS phase

30 The **PROCESS** phase includes calculating in parallel the sixteen 2-D DCTs by processing the eight-component

vectors l, m, \dots, s as shown in Fig. 5. It is noted for example, that the coefficients of the 2-D DCT of the block $(0,3)$ will constitute the third component of the vectors a, b, c, d of Fig. 3.

5 ORDER phase

The ORDER phase includes arranging the output sequences of the eight 2-D DCTs in eight vectors l', m', \dots, s' thus defined:

$$\begin{array}{lll}
 10 & l' = \begin{bmatrix} a[0] \\ b[0] \\ c[0] \\ d[0] \\ a[1] \\ b[1] \\ c[1] \\ d[1] \end{bmatrix}, & m' = \begin{bmatrix} a[2] \\ b[2] \\ c[2] \\ d[2] \\ a[3] \\ b[3] \\ c[3] \\ d[3] \end{bmatrix}, & n' = \begin{bmatrix} a[4] \\ b[4] \\ c[4] \\ d[4] \\ a[5] \\ b[5] \\ c[5] \\ d[5] \end{bmatrix}, \\
 15 & & & \\
 20 & o' = \begin{bmatrix} a[6] \\ b[6] \\ c[6] \\ d[6] \\ a[7] \\ b[7] \\ c[7] \\ d[7] \end{bmatrix}, & p' = \begin{bmatrix} e[0] \\ f[0] \\ g[0] \\ h[0] \\ e[1] \\ f[1] \\ g[1] \\ h[1] \end{bmatrix}, & q' = \begin{bmatrix} e[2] \\ f[2] \\ g[2] \\ h[2] \\ e[3] \\ f[3] \\ g[3] \\ h[3] \end{bmatrix}, \\
 25 & & & \\
 30 & r' = \begin{bmatrix} e[4] \\ f[4] \\ g[4] \\ h[4] \\ e[5] \\ f[5] \\ g[5] \\ h[5] \end{bmatrix}, & s' = \begin{bmatrix} e[6] \\ f[6] \\ g[6] \\ h[6] \\ e[7] \\ f[7] \\ g[7] \\ h[7] \end{bmatrix}, &
 \end{array}$$

It is noted, for example, that the coefficients of the 2-D DCT of the block $(0,3)$ will constitute the

components 4, 5, 6, 7 of the vector \mathbf{m}' .

OUTPUT phase

This phase includes rearranging the output data: starting from the eight-component vectors \mathbf{a} , \mathbf{b} , ..., \mathbf{h} , a 64 component vector defined as follows, is constructed:

$$\begin{bmatrix} y[0] & y[1] & \dots & y[7] \\ \vdots & \vdots & \vdots & \vdots \\ y[54] & y[55] & \dots & y[63] \end{bmatrix} = \begin{bmatrix} l[0] & l[1] & l[4] & l[5] & m[0] & m[1] & m[4] & m[5] \\ l[2] & l[3] & l[6] & l[7] & m[2] & m[3] & m[6] & m[7] \\ n[0] & n[1] & n[4] & n[5] & o[0] & o[1] & o[4] & o[5] \\ n[2] & n[3] & n[6] & n[7] & o[2] & o[3] & o[6] & o[7] \\ p[0] & p[1] & p[4] & p[5] & q[0] & q[1] & q[4] & q[5] \\ p[2] & p[3] & p[6] & p[7] & q[2] & q[3] & q[6] & q[7] \\ r[0] & r[1] & r[4] & r[5] & s[0] & s[1] & s[4] & s[5] \\ r[2] & r[3] & r[6] & r[7] & s[2] & s[3] & s[6] & s[7] \end{bmatrix} \quad (7)$$

Parallel computation of four 4x4 DCTs

For $N = 4$, equation (5) becomes

$$y_{m,n} = \sum_{i=0}^3 \sum_{j=0}^3 x_{i,j} \cos\left(\frac{(2i+1)m}{8}\pi\right) \cos\left(\frac{(2j+1)n}{8}\pi\right) \quad (8)$$

If:

$$Y_{16 \times 1} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix}, \quad F_{64 \times 1} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix}$$

where:

$$\begin{aligned} Y_1 &= [Y_{1,0}, Y_{1,1}, Y_{1,2}, Y_{1,3}]^T \\ \{f_{0,r}\}_{r=0}^3 &= DCT(\{A_{1,i}\}_{i=0}^3), \quad \{f_{2,r}\}_{r=0}^3 = DCT(\{B_{3,i}\}_{i=0}^3) \\ \{f_{1,r}\}_{r=0}^3 &= DCT(\{A_{3,i}\}_{i=0}^3), \quad \{f_{3,r}\}_{r=0}^3 = DCT(\{B_{1,i}\}_{i=0}^3) \\ \{A_{1,i}\}_{i=0}^3 &= \{x_{0,0}, x_{1,1}, x_{2,2}, x_{3,3}\}, \\ \{A_{3,i}\}_{i=0}^3 &= \{x_{1,0}, x_{3,1}, x_{0,2}, x_{2,3}\}, \\ \{B_{3,i}\}_{i=0}^3 &= \{x_{2,0}, x_{0,1}, x_{3,2}, x_{1,3}\}, \end{aligned}$$

it may be demonstrated that:

$$Y_{16*1} = (E_4)_{16} F_{16*1} \quad (9)$$

where:

$$(E_4)_{16} = \begin{bmatrix} (H_0)_4 & (H_0)_4 & (H_0)_4 & (H_0)_4 \\ (H_1)_4 & (H_3)_4 & -(H_3)_4 & -(H_1)_4 \\ (H_2)_4 & -(H_2)_4 & -(H_2)_4 & (H_2)_4 \\ (H_3)_4 & -(H_1)_4 & (H_1)_4 & -(H_3)_4 \end{bmatrix} \quad (10)$$

The matrices $(H_i)_4$, $i = 0, 1, 2, 3$ are as follows:

$$(H_0)_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (H_1)_4 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & 0 \end{bmatrix},$$

$$(H_2)_4 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & -\frac{1}{2} \end{bmatrix}, \quad (H_3)_4 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & -\frac{1}{2} \\ \frac{1}{2} & 0 & -\frac{1}{2} & 0 \end{bmatrix}.$$

The monodimensional DCT, or briefly the 1-D DCT, is expressed by the matrix $(1\text{-D DCT})_4$ given by:

$$C_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ C_8^1 & C_8^3 & -C_8^3 & -C_8^1 \\ C_4^1 & -C_4^1 & -C_4^1 & C_4^1 \\ C_8^3 & -C_8^1 & C_8^1 & -C_8^3 \end{bmatrix} \quad \text{where } C_n^m = \cos\left(\frac{m}{n}\pi\right)$$

From these equations it may be said that the computation of one 4x4 DCT may be divided into two steps:

computation of four 1-D DCT, each performed on an appropriate sequence of four pixels.

computation of the 2-D DCT starting from the four 1-D DCT.

These two steps are carried out in a similar manner, and are implemented with the same hardware that is used

twice. Let us consider now how to calculate in parallel four 4x4 DCTs. The total 64 samples are obtained from the 4x4 blocks into which an 8x8 block is subdivided. The procedure is subdivided in distinct phases to each of which corresponds an architectural block. A whole view is shown in Fig. 6. This figure highlights the transformations carried out on each 4x4 block.

INPUT phase

The pixel of each quadrant (i,j) , $0 \leq i,j \leq 1$ are ordered to constitute the vectors:

$$A_1^{i,j} = \begin{bmatrix} x_{0,0}^{i,j} \\ x_{1,1}^{i,j} \\ x_{2,2}^{i,j} \\ x_{3,3}^{i,j} \end{bmatrix}, \quad A_3^{i,j} = \begin{bmatrix} x_{1,0}^{i,j} \\ x_{3,1}^{i,j} \\ x_{0,2}^{i,j} \\ x_{2,3}^{i,j} \end{bmatrix}, \quad B_3^{i,j} = \begin{bmatrix} x_{2,0}^{i,j} \\ x_{0,1}^{i,j} \\ x_{3,2}^{i,j} \\ x_{1,3}^{i,j} \end{bmatrix}, \quad B_1^{i,j} = \begin{bmatrix} x_{3,0}^{i,j} \\ x_{2,1}^{i,j} \\ x_{1,2}^{i,j} \\ x_{0,3}^{i,j} \end{bmatrix}$$

After arranging the data in 16 four-component vectors, we define the eight-component vectors l , m , n , o , constituted by the first, second, third and fourth components, respectively, of the initial vectors constituted by the pixels of the 00 and 01 quadrants, and the p , q , r , s , vectors constituted by the first, second, third and fourth components, respectively, of the initial vectors constituted by the pixels of the 10 and 11 quadrants. Precisely:

$$l = \begin{bmatrix} A_1[0]^{0,0} \\ A_3[0]^{0,0} \\ B_3[0]^{0,0} \\ B_1[0]^{0,0} \\ A_1[0]^{0,1} \\ A_3[0]^{0,1} \\ B_3[0]^{0,1} \\ B_1[0]^{0,1} \end{bmatrix}, \quad m = \begin{bmatrix} A_1[1]^{0,0} \\ A_3[1]^{0,0} \\ B_3[1]^{0,0} \\ B_1[1]^{0,0} \\ A_1[1]^{0,1} \\ A_3[1]^{0,1} \\ B_3[1]^{0,1} \\ B_1[1]^{0,1} \end{bmatrix}, \quad n = \begin{bmatrix} A_1[2]^{0,0} \\ A_3[2]^{0,0} \\ B_3[2]^{0,0} \\ B_1[2]^{0,0} \\ A_1[2]^{0,1} \\ A_3[2]^{0,1} \\ B_3[2]^{0,1} \\ B_1[2]^{0,1} \end{bmatrix}, \quad o = \begin{bmatrix} A_1[3]^{0,0} \\ A_3[3]^{0,0} \\ B_3[3]^{0,0} \\ B_1[3]^{0,0} \\ A_1[3]^{0,1} \\ A_3[3]^{0,1} \\ B_3[3]^{0,1} \\ B_1[3]^{0,1} \end{bmatrix}$$

$$\begin{aligned}
 p &= \begin{bmatrix} A_1[0]^{1,0} \\ A_3[0]^{1,0} \\ B_3[0]^{1,0} \\ B_1[0]^{1,0} \\ A_1[0]^{1,1} \\ A_3[0]^{1,1} \\ B_3[0]^{1,1} \\ B_1[0]^{1,1} \end{bmatrix}, & p &= \begin{bmatrix} A_1[1]^{1,0} \\ A_3[1]^{1,0} \\ B_3[1]^{1,0} \\ B_1[1]^{1,0} \\ A_1[1]^{1,1} \\ A_3[1]^{1,1} \\ B_3[1]^{1,1} \\ B_1[1]^{1,1} \end{bmatrix}, & q &= \begin{bmatrix} A_1[2]^{1,0} \\ A_3[2]^{1,0} \\ B_3[2]^{1,0} \\ B_1[2]^{1,0} \\ A_1[2]^{1,1} \\ A_3[2]^{1,1} \\ B_3[2]^{1,1} \\ B_1[2]^{1,1} \end{bmatrix}, & s &= \begin{bmatrix} A_1[3]^{1,0} \\ A_3[3]^{1,0} \\ B_3[3]^{1,0} \\ B_1[3]^{1,0} \\ A_1[3]^{1,1} \\ A_3[3]^{1,1} \\ B_3[3]^{1,1} \\ B_1[3]^{1,1} \end{bmatrix}
 \end{aligned}$$

By taking into account the way in which the vectors $A_1^{i,j}$, $A_3^{i,j}$, $B_3^{i,j}$, $B_1^{i,j}$ are defined, the arrangement detailed in Fig. 7 is obtained. It should be noted that in this figure the original 8x8 block is subdivided in the four 4x4 quadrants, within each quadrant (i,j) , the pixels belonging to the respective vectors $A_1^{i,j}$, $A_3^{i,j}$, $B_3^{i,j}$, $B_1^{i,j}$ have different shadings in the figure.

According to what has been described above, the computation of a 4x4 DCT may be subdivided in two stages: consequently, the **PROCESS** phase that is the only phase in which arithmetical operations are performed, is done twice:

a first time, to compute in parallel the sixteen 1-D DCTs;

a second time, to compute in parallel four 4x4 DCT starting from the coefficients of the 1-D DCTs.

The variable *stage* indicates whether the first or second calculation stage is being performed.

During the **INPUT** phase the variable *stage* is updated to the value 0.

At the input in the **PROCESS** phase, there are 64 input MUXes that are controlled by the variable *stage*. Each MUX receives two inputs:

a pixel of the original picture, coming from the **INPUT** phase (this input is selected when *stage* = 0);

a coefficient of a 1-D DCT, coming from the **ORDER**

phase (this input is selected when *stage* = 1).

PROCESS phase

This phase includes processing the *l*, *m*, ..., *s* vectors as shown in Fig. 8. In this figure the following symbols are used:

$$A = \left\{ \begin{array}{ll} 2C_8^1 \times I_{8 \times 8} & \text{per stage} = 0 \\ \begin{bmatrix} (H_1)_4 & 0 \\ 0 & (H_1)_4 \end{bmatrix} & \text{per stage} = 1 \end{array} \right\} \quad (11)$$

$$B = \left\{ \begin{array}{ll} 2C_8^3 \times I_{8 \times 8} & \text{per stage} = 0 \\ \begin{bmatrix} -(H_3)_4 & 0 \\ 0 & -(H_3)_4 \end{bmatrix} & \text{per stage} = 1 \end{array} \right\} \quad (12)$$

$$C = \left\{ \begin{array}{ll} 2C_4^1 \times I_{8 \times 8} & \text{per stage} = 0 \\ \begin{bmatrix} (H_2)_4 & 0 \\ 0 & (H_2)_4 \end{bmatrix} & \text{per stage} = 1 \end{array} \right\} \quad (13)$$

$$t = \left\{ \begin{array}{ll} 1 & \text{per stage} = 0 \\ 2 & \text{per stage} = 1 \end{array} \right\} \quad (14)$$

At the output of the **PROCESS** structure there are 64 DEMUXes controlled by the variable *stage*. The DEMUX address the data according to two conditions:

if *stage* = 0, the input datum to each DEMUX is a coefficient of a 1-D DCT; therefore the datum must be further processed and, for this purpose, is conveyed to the **ORDER** phase;

if *stage* = 1, the input datum to each DEMUX is a coefficient of a 2-D DCT; therefore the datum must not be processed further and therefore is conveyed to the **OUTPUT** phase.

ORDER phase

The **ORDER** phase includes arranging the output sequence of the eight 1-D DCTs in eight *l'*, *m'*, ..., *s'*

vectors, thus defined:

$$\begin{aligned}
 & \begin{matrix} 5 \\ 10 \end{matrix} \quad l' = \begin{bmatrix} f_0^{0,0} \\ f_0^{0,1} \end{bmatrix} = \begin{bmatrix} a[0] \\ b[0] \\ c[0] \\ d[0] \\ a[4] \\ b[4] \\ c[4] \\ d[4] \end{bmatrix}, \quad m' = \begin{bmatrix} f_1^{0,0} \\ f_1^{0,1} \end{bmatrix} = \begin{bmatrix} a[1] \\ b[1] \\ c[1] \\ d[1] \\ a[5] \\ b[5] \\ c[5] \\ d[5] \end{bmatrix}, \\
 & \begin{matrix} 15 \\ 20 \end{matrix} \quad n' = \begin{bmatrix} f_2^{0,0} \\ f_2^{0,1} \end{bmatrix} = \begin{bmatrix} a[2] \\ b[2] \\ c[2] \\ d[2] \\ a[6] \\ b[6] \\ c[6] \\ d[6] \end{bmatrix}, \quad o' = \begin{bmatrix} f_3^{0,0} \\ f_3^{0,1} \end{bmatrix} = \begin{bmatrix} a[3] \\ b[3] \\ c[3] \\ d[3] \\ a[7] \\ b[7] \\ c[7] \\ d[7] \end{bmatrix}, \\
 & \begin{matrix} 25 \\ 30 \end{matrix} \quad p' = \begin{bmatrix} f_0^{1,0} \\ f_0^{1,1} \end{bmatrix} = \begin{bmatrix} e[0] \\ f[0] \\ g[0] \\ h[0] \\ e[4] \\ f[4] \\ g[4] \\ h[4] \end{bmatrix}, \quad q' = \begin{bmatrix} f_1^{1,0} \\ f_1^{1,1} \end{bmatrix} = \begin{bmatrix} e[1] \\ f[1] \\ g[1] \\ h[1] \\ e[5] \\ f[5] \\ g[5] \\ h[5] \end{bmatrix}, \\
 & \quad r' = \begin{bmatrix} f_2^{1,0} \\ f_2^{1,1} \end{bmatrix} = \begin{bmatrix} e[2] \\ f[2] \\ g[2] \\ h[2] \\ e[6] \\ f[6] \\ g[6] \\ h[6] \end{bmatrix}, \quad s' = \begin{bmatrix} f_3^{1,0} \\ f_3^{1,1} \end{bmatrix} = \begin{bmatrix} e[3] \\ f[3] \\ g[3] \\ h[3] \\ e[7] \\ f[7] \\ g[7] \\ h[7] \end{bmatrix}
 \end{aligned}$$

35 After the **ORDER** phase the variable stage is updated

to the value 1. The output data from the **ORDER** phase are sent to the **PROCESS** phase.

OUTPUT phase

This phase includes rearranging the data originating from the second ($stage = 1$) execution of the **PROCESS** step: starting from these data, which constitute the eight-component vectors: a, b, \dots, h , the output block $Y_{N \times N}$ is thus

defined:

$$\begin{bmatrix} y[0] & y[1] & \dots & y[7] \\ \vdots & \vdots & \vdots & \vdots \\ y[54] & y[55] & \dots & y[63] \end{bmatrix} = \begin{bmatrix} a[0] & b[0] & c[0] & d[0] & e[0] & f[0] & g[0] & h[0] \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a[3] & b[3] & c[3] & d[3] & e[3] & f[3] & g[3] & h[3] \\ e[4] & f[4] & g[4] & h[4] & a[4] & b[4] & c[4] & d[4] \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ e[7] & f[7] & g[7] & h[7] & a[7] & b[7] & c[7] & d[7] \end{bmatrix} \quad (15)$$

The main differences between the hardware for calculating the four 4×4 DCTs and the hardware needed for the sixteen 2×2 DCTs are the following:

the ordering sequences of the pixels of the block of the original picture depend on the chosen DCT size; to execute the sixteen 2×2 DCTs the **PROCESS** step must be carried out only once; instead, to execute the four 4×4 DCTs the **PROCESS** step must be repeated two times;

the operations executed during the **PROCESS** phase are not always the same for the two cases.

Computation of an 8×8 DCT

For $N = 8$ equation (5) becomes:

$$y_{m,n} = \sum_{i=0}^7 \sum_{j=0}^7 x_{i,j} \cos \left[\frac{(2i+1)m}{16} \pi \right] \cos \left[\frac{(2j+1)n}{16} \pi \right] \quad (16)$$

Putting:

$$Y_{64 \times 1} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_7 \end{bmatrix}, \quad F_{64 \times 1} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_7 \end{bmatrix}$$

5 where:

$$y_i = [y_{i,0} \quad y_{i,1} \quad \cdots \quad y_{i,7}]$$

$$\begin{aligned} \{f_{0,r}\}_{r=0}^7 &= DCT(\{A_{1,i}\}_{i=0}^7), \quad \{f_{4,r}\}_{r=0}^7 = DCT(\{B_{7,i}\}_{i=0}^7), \\ \{f_{1,r}\}_{r=0}^7 &= DCT(\{A_{3,i}\}_{i=0}^7), \quad \{f_{5,r}\}_{r=0}^7 = DCT(\{B_{5,i}\}_{i=0}^7), \\ \{f_{2,r}\}_{r=0}^7 &= DCT(\{A_{5,i}\}_{i=0}^7), \quad \{f_{6,r}\}_{r=0}^7 = DCT(\{B_{3,i}\}_{i=0}^7), \\ \{f_{3,r}\}_{r=0}^7 &= DCT(\{A_{7,i}\}_{i=0}^7), \quad \{f_{7,r}\}_{r=0}^7 = DCT(\{B_{1,i}\}_{i=0}^7), \\ \{A_{1,i}\}_{i=0}^7 &= \{x_{0,0} \quad x_{1,1} \quad x_{2,2} \quad x_{3,3} \quad x_{4,4} \quad x_{5,5} \quad x_{6,6} \quad x_{7,7}\}, \\ \{A_{3,i}\}_{i=0}^7 &= \{x_{1,0} \quad x_{4,1} \quad x_{7,2} \quad x_{5,3} \quad x_{2,4} \quad x_{0,5} \quad x_{3,6} \quad x_{6,7}\}, \\ \{A_{5,i}\}_{i=0}^7 &= \{x_{2,0} \quad x_{7,1} \quad x_{3,2} \quad x_{1,3} \quad x_{6,4} \quad x_{4,5} \quad x_{0,6} \quad x_{5,7}\}, \\ \{A_{7,i}\}_{i=0}^7 &= \{x_{3,0} \quad x_{5,1} \quad x_{1,2} \quad x_{7,3} \quad x_{0,4} \quad x_{6,5} \quad x_{2,6} \quad x_{4,7}\}, \\ \{B_{7,i}\}_{i=0}^7 &= \{x_{4,0} \quad x_{2,1} \quad x_{6,2} \quad x_{0,3} \quad x_{7,4} \quad x_{1,5} \quad x_{5,6} \quad x_{3,7}\}, \\ \{B_{5,i}\}_{i=0}^7 &= \{x_{5,0} \quad x_{0,1} \quad x_{4,2} \quad x_{6,3} \quad x_{1,4} \quad x_{3,5} \quad x_{7,6} \quad x_{2,7}\}, \\ \{B_{3,i}\}_{i=0}^7 &= \{x_{6,0} \quad x_{3,1} \quad x_{0,2} \quad x_{2,3} \quad x_{5,4} \quad x_{7,5} \quad x_{4,6} \quad x_{1,7}\}, \\ \{B_{1,i}\}_{i=0}^7 &= \{x_{7,0} \quad x_{6,1} \quad x_{5,2} \quad x_{4,3} \quad x_{3,4} \quad x_{2,5} \quad x_{1,6} \quad x_{0,7}\} \end{aligned}$$

it may be demonstrated that:

$$Y_{64 \times 1} = (E_8)_{64} F_{64 \times 1} \quad (17)$$

where:

$$\begin{aligned}
5 \quad (E_8)_{64} = & \begin{bmatrix} (H_0)_8 & (H_0)_8 & (H_0)_8 & (H_0)_8 & (H_0)_8 & (H_0)_8 & (H_0)_8 & (H_0)_8 \\ (H_1)_8 & (H_3)_8 & (H_5)_8 & (H_7)_8 & -(H_7)_8 & -(H_5)_8 & -(H_3)_8 & -(H_1)_8 \\ (H_2)_8 & (H_6)_8 & -(H_6)_8 & -(H_2)_8 & -(H_2)_8 & -(H_6)_8 & (H_6)_8 & (H_2)_8 \\ (H_3)_8 & -(H_7)_8 & -(H_1)_8 & -(H_5)_8 & (H_5)_8 & (H_1)_8 & (H_7)_8 & -(H_3)_8 \\ (H_4)_8 & -(H_4)_8 & -(H_4)_8 & (H_4)_8 & (H_4)_8 & -(H_4)_8 & -(H_4)_8 & (H_4)_8 \\ (H_5)_8 & -(H_1)_8 & (H_7)_8 & (H_3)_8 & -(H_3)_8 & -(H_7)_8 & (H_1)_8 & -(H_5)_8 \\ (H_6)_8 & -(H_2)_8 & (H_2)_8 & -(H_6)_8 & -(H_6)_8 & (H_2)_8 & -(H_2)_8 & (H_6)_8 \\ (H_7)_8 & -(H_5)_8 & (H_3)_8 & -(H_1)_8 & (H_1)_8 & -(H_3)_8 & (H_5)_8 & -(H_7)_8 \end{bmatrix} \quad (18)
\end{aligned}$$

the matrices $(H_i)_8$ $i = 0, 1, \dots, 7$ have the following expressions:

$$\begin{aligned}
15 \quad (H_0)_8 = & \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},
\end{aligned}$$

$$\begin{aligned}
20 \quad (H_1)_8 = & \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \end{bmatrix},
\end{aligned}$$

$$\begin{array}{l}
 5 \\
 10 \\
 15 \\
 20 \\
 25
 \end{array}
 \begin{array}{l}
 (H_2)_8 = \\
 \\
 \\
 (H_3)_8 = \\
 \\
 \end{array}
 \begin{bmatrix}
 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\
 \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\
 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\
 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 \\
 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} \\
 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & -\frac{1}{2}
 \end{bmatrix},
 \begin{bmatrix}
 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\
 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\
 \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\
 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\
 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} \\
 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & -\frac{1}{2} & 0
 \end{bmatrix},
 \end{array}$$

$$\begin{array}{rcl}
 & & (H_4)_8 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 \end{bmatrix}, \\
 5 & & \\
 10 & & \\
 15 & & (H_5)_8 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 & 0 \end{bmatrix}, \\
 20 & & \\
 25 & &
 \end{array}$$

$$(H_6)_8 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 & 0 \\ 0 & -\frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$(H_7)_8 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & -\frac{1}{2} \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The 1-D DCT is expressed by the matrix:

$$(1-D DCT)_8 = C_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ C_{16}^1 & C_{16}^3 & C_{16}^5 & C_{16}^7 & -C_{16}^7 & -C_{16}^5 & -C_{16}^3 & -C_{16}^1 \\ C_8^3 & C_8^3 & C_8^5 & C_8^7 & C_8^7 & C_8^5 & C_8^3 & C_8^1 \\ C_{16}^3 & -C_{16}^7 & -C_{16}^1 & -C_{16}^5 & C_{16}^5 & C_{16}^1 & C_{16}^7 & -C_{16}^3 \\ C_4^1 & -C_4^1 & -C_4^1 & C_4^1 & C_4^1 & -C_4^1 & -C_4^1 & C_4^1 \\ C_{16}^5 & -C_{16}^1 & C_{16}^7 & C_{16}^3 & -C_{16}^3 & -C_{16}^7 & C_{16}^1 & -C_{16}^5 \\ C_8^3 & -C_8^1 & C_8^1 & C_8^5 & C_8^5 & C_8^1 & C_8^3 & C_8^3 \\ C_{16}^7 & -C_{16}^5 & C_{16}^3 & -C_{16}^1 & C_{16}^1 & -C_{16}^3 & -C_{16}^5 & -C_{16}^7 \end{bmatrix}$$

Where we put:

$$C_n^m = \cos\left(\frac{m}{n}\pi\right)$$

From the above equations it is evident that the computation of an 8x8 DCT may be subdivided in two stages:

- 5 calculating eight 1-D DCTs, each for a certain sequence of eight pixels;
- calculating the 2-D DCT, starting from the eight 1-D DCTs.

10 These two stages may be executed through the same hardware using it twice. The processing is subdivided in different steps, to each of which corresponds an architectural block. A whole view of the hardware is shown in Fig. 9.

INPUT phase

- 15 The pixels of the 8x8 input block are ordered to constitute the eight-component vectors l, m, n, o, p, q, r, s :

$$20 \quad l = \begin{bmatrix} A_1[0] \\ A_3[0] \\ A_5[0] \\ A_7[0] \\ B_7[0] \\ B_5[0] \\ B_3[0] \\ B_1[0] \end{bmatrix}, \quad m = \begin{bmatrix} A_1[1] \\ A_3[1] \\ A_5[1] \\ A_7[1] \\ B_7[1] \\ B_5[1] \\ B_3[1] \\ B_1[1] \end{bmatrix}, \quad \dots, \quad s = \begin{bmatrix} A_1[7] \\ A_3[7] \\ A_5[7] \\ A_7[7] \\ B_7[7] \\ B_5[7] \\ B_3[7] \\ B_1[7] \end{bmatrix}$$

- 25 By taking into account the way in which the vectors $A_1, A_3, A_5, A_7, B_7, B_5, B_3, B_1$ are defined, we obtain the detailed arrangement of Fig. 10. It should be noticed that in this figure the pixels belonging to the vectors $A_1, A_3, A_5, A_7, B_7, B_5, B_3, B_1$ are countersigned by
- 30 different shadings.

As shown above, the computation of an 8x8 DCT may be subdivided into two stage. The **PROCESS** step, which is the only phase in which mathematical operations are

performed, is performed twice:

the first time, to compute in parallel sixteen 1-D DCTs;

the second time, to compute the 8x8 DCT starting from the coefficients of the sixteen 1-D DCTs.

The variable *stage* indicates whether the first or second calculation step is being performed. During the **INPUT** phase, the variable *stage* is updated to the value 0.

At the input of the **PROCESS** structure, there are 64 MUXes controlled by the variable *stage*. Each MUX receives two inputs:

a pixel of the original picture, originating from the **INPUT** phase (this input is selected when *stage* = 0);

a coefficient of a 1-D DCT, originating from the **ORDER** phase (this input is selected when *stage* = 1).

PROCESS phase

This phase includes processing the $1, m, \dots, s$ vectors as shown in Fig. 11. In this figure, the following symbols are used:

$$A = \begin{cases} 2C_8^1 \times I_{8 \times 8} & \text{per stage} = 0 \\ (H_2)_8 & \text{per stage} = 1 \end{cases} \quad (19)$$

$$B = \begin{cases} 2C_8^3 \times I_{8 \times 8} & \text{per stage} = 0 \\ -(H_6)_8 & \text{per stage} = 1 \end{cases} \quad (20)$$

$$C = \begin{cases} 2C_4^1 \times I_{8 \times 8} & \text{per stage} = 0 \\ (H_4)_8 & \text{per stage} = 1 \end{cases} \quad (21)$$

$$t = \begin{cases} 1 & \text{per stage} = 0 \\ 2 & \text{per stage} = 1 \end{cases} \quad (22)$$

At the output of the **PROCESS** structure there are 64 DEMUXes controlled by the variable *stage*. The DEMUXes address the data according to two possibilities:

if $stage = 0$, the input datum to each DEMUX is a coefficient of a 1-D DCT; therefore, the datum must be further processed and, for this purpose, is sent to the **ORDER** phase;

if $stage = 1$, the input datum to each DEMUX is a coefficient of a 2-D DCT; therefore, the datum does not need any further processing and therefore is sent to the **OUTPUT** phase.

10 ORDER phase

This phase includes arranging the output sequence of the eight 1-D DCTs in eight l', m', \dots, s' vectors, thus defined:

$$15 \quad l' = f_0 = \begin{bmatrix} a[0] \\ b[0] \\ c[0] \\ d[0] \\ e[0] \\ f[0] \\ g[0] \\ h[0] \end{bmatrix}, \quad m' = f_1 = \begin{bmatrix} a[1] \\ b[1] \\ c[1] \\ d[1] \\ e[1] \\ f[1] \\ g[1] \\ h[1] \end{bmatrix}, \dots, \quad q' = f_7 = \begin{bmatrix} a[7] \\ b[7] \\ c[7] \\ d[7] \\ e[7] \\ f[7] \\ g[7] \\ h[7] \end{bmatrix}$$

Following the **ORDER** phase the variable $stage$ is updated to the value 1. The output data from the **ORDER** phase are sent to the **PROCESS** phase.

OUTPUT phase

This phase includes rearranging the data originating from the second execution of the **PROCESS** step (that is, with $stage = 1$): starting from these data, which constitute the eight-component vectors a, b, \dots, h , the output block $Y_{N \times N}$ defined as follows is constituted:

$$30 \quad \begin{bmatrix} y[0] & y[1] & \dots & y[7] \\ \vdots & \vdots & \vdots & \vdots \\ y[54] & y[55] & \vdots & y[63] \end{bmatrix} = \begin{bmatrix} a[0] & b[0] & c[0] & d[0] & e[0] & f[0] & g[0] & h[0] \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a[7] & b[7] & c[7] & d[7] & e[7] & f[7] & g[7] & h[7] \end{bmatrix} \quad (23)$$

The main differences between the hardware that

calculates an 8x8 DCT and the hardware that calculates the four 4x4 DCTs are:

the sequences into which must be arranged the pixels of a block of the original picture depend on the chosen size of the DCT;

the operations executed during the PROCESS step are not always the same for the two cases.

Procedure for calculating the DCT for blocks of scaleable size (8x8 DCTs, 4x4 DCTs and 2x2 DCTs)

From the above described procedures, an algorithm for calculating a chosen one of 8x8 DCT or four 4x4 DCTs (in parallel) or sixteen 2x2 DCTs (in parallel) may be derived. The selection is made by the user by assigning a certain value to the global variable *size*:

$$\text{size} = \begin{cases} 0 & \text{for an 8x8 DCT} \\ 1 & \text{for four DCTs} \\ 2 & \text{for sixteen 2x2 DCTs} \end{cases}$$

The procedure is subdivided in various phases (regardless of the value of the variable *size*), to each of which corresponds an architectural block. A whole view is shown in Fig. 12. Each phase has been organized in order to provide for partial results corresponding to the chosen value, minimizing redundancies. Sometimes the operations performed are different depending on the value of *size*. In these cases, the architecture considers a MUX whose control input is *size*. Let us examine now the various phases and highlight the differences in respect to the architectures that have already been described above:

INPUT phase

The object of this phase, depicted in Fig. 13, is to arrange the data to allow the computation starting from the arranged data of the 1-D DCTs. This is done by inputting the luminance values of the pixels (8x8 matrix) and arranging them in eight-component vectors *l*,

$m, \dots, s.$

For example:

$$l[0] = x_{0,0}$$

$$l[1] = \begin{cases} x_{1,2} & \text{per size} = 0 \text{ or } 1 \\ x_{2,0} & \text{per size} = 2 \end{cases}$$

$$s[7] = \begin{cases} x_{0,7} & \text{per size} = 0 \\ x_{4,7} & \text{per size} = 1 \\ x_{7,7} & \text{per size} = 2 \end{cases}$$

10 PROCESS phase with stage = 0

This phase includes calculating in parallel the eight 1-D DCTs by processing the vectors l, m, \dots, s as shown in Fig. 14. In this figure may be observed the use of 16 MUXes controlled by the variable *size*. The eight MUXes on the left serve to bypass the operations required for the computation of the 8x8 DCT. Thus, the bypass occurs when *size* = 1 or 2, while it does not occur for *size* = 0. The eight MUXes on the right serve to output only the result that corresponds to the pre-selected value of *size*.

$$t = \begin{cases} 1 & \text{per stage} = 0 \\ 2 & \text{per stage} = 1 \end{cases} \quad (24)$$

$$A = \begin{cases} 2C_8^1 \times I_{8 \times 8} & \text{per stage, size} = (0,0) \text{ or } (0,1) \\ I_{8 \times 8} & \text{per stage, size} = (0,2) \text{ or } (1,2) \\ (H_2)_8 & \text{per stage, size} = (1,0) \\ \begin{bmatrix} (H_1)_4 & 0 \\ 0 & (H_1)_4 \end{bmatrix} & \text{per stage, size} = (1,1) \end{cases} \quad (25)$$

$$B = \begin{cases} 2C_8^3 \times I_{8 \times 8} & \text{per stage, size} = (0,0) \text{ or } (0,1) \\ I_{8 \times 8} & \text{per stage, size} = (0,2) \text{ or } (1,2) \\ -(H_6)_8 & \text{per stage, size} = (1,0) \\ \begin{bmatrix} -(H_3)_4 & 0 \\ 0 & -(H_3)_4 \end{bmatrix} & \text{per stage, size} = (1,1) \end{cases} \quad (26)$$

$$C = \left\{ \begin{array}{ll} 2C_4^1 \times I_{8 \times 8} & \text{per stage, size} = (0,0) \text{ or } (0,1) \\ I_{8 \times 8} & \text{per stage, size} = (0,2) \text{ or } (1,2) \\ (H_4)_8 & \text{per stage, size} = (1,0) \\ \left[\begin{array}{cc} (H_2)_4 & 0 \\ 0 & (H_2)_4 \end{array} \right] & \text{per stage, size} = (1,1) \end{array} \right\} \quad (27)$$

$$D = \left\{ \begin{array}{ll} 2C_{16}^1 \times I_{8 \times 8} & \text{per stage} = 0 \\ (H_1)_8 & \text{per stage} = 1 \end{array} \right\} \quad (28)$$

$$E = \left\{ \begin{array}{ll} 2C_8^3 \times I_{8 \times 8} & \text{per stage} = 0 \\ (H_5)_8 & \text{per stage} = 1 \end{array} \right\} \quad (29)$$

$$F = \left\{ \begin{array}{ll} 2C_{16}^7 \times I_{8 \times 8} & \text{per stage} = 0 \\ -(H_7)_8 & \text{per stage} = 1 \end{array} \right\} \quad (30)$$

$$G = \left\{ \begin{array}{ll} 2C_{16}^5 \times I_{8 \times 8} & \text{per stage} = 0 \\ -(H_3)_8 & \text{per stage} = 1 \end{array} \right\} \quad (31)$$

The scheme in Fig. 14 may be subdivided into the architectural blocks shown in Fig. 15. For example, two vectors each of eight components (each component being a pixel, that may have been processed already) are input to the QA block, which outputs two vectors of eight components: the first vector is the sum of the two input vectors, while the second vector is the difference between the two input vectors that is successively processed with the linear operator A. It should be noted that the operators A, B, C, D, E, F, G are 8x8 matrices.

By considering a lower level of generalization, the QA, QB, QC blocks are shown in detail in Figures 16, 17 and 18, respectively. In these figures the MUXes are controlled by three bits, which correspond to the variable *stage* (which may take the value 0 or 1, and thus is represented by a bit) and the variable *size* (which may take the value 0, 1 or 2, and thus is represented by two bits). The blocks QD, QE, QF, QG are shown in detail in Figures 19, 20, 21 and 22,

respectively. In these figures the MUXes are controlled by a bit that corresponds to the variable *stage*.

ORDER phase

The ORDER phase, depicted in Fig. 23, includes arranging the output sequences of the eight 1-D DCTs in eight vectors l' , m' , ..., s' . For example:

$$l'[0] = a[0];$$

$$l'[1] = b[0];$$

$$\vdots$$

$$l'[4] = \begin{cases} e[0] & \text{per size} = 0 \\ a[4] & \text{per size} = 1 \\ a[1] & \text{per size} = 2 \end{cases}$$

$$\vdots$$

$$s'[7] = h[7];$$

OUTPUT phase

This phase, depicted in Fig. 24, includes rearranging the data coming from the second (that is with *stage* = 1) execution of the PROCESS step. Starting from these data, constituting the eight-component vectors a , b , ..., h , the output block $y_{N \times N}$ is constituted.

For example:

$$y[0] = \begin{cases} a[0] & \text{per size} = 0 \text{ or } 1 \\ l[2] & \text{per size} = 2 \end{cases}$$

$$y[1] = \begin{cases} b[0] & \text{per size} = 0 \text{ or } 1 \\ l[1] & \text{per size} = 2 \end{cases}$$

$$\vdots$$

$$y[63] = \begin{cases} h[7] & \text{per size} = 0 \\ d[7] & \text{per size} = 1 \\ s[7] & \text{per size} = 2 \end{cases}$$

Description of the Drawings

A functional block diagram of a picture

Description of the Drawings

A functional block diagram of a picture compressor-coder according to the present invention may be represented as shown in Figure 1.

Essentially, the compressor-coder performs a hybrid compression based on a fractal coding in the DCT domain. This is made possible by the peculiar architecture of parallel calculation of the DCT on blocks of scaleable size of pixels, as described above.

Hereinbelow, the remaining figures are described one by one:

Figure 2 is a flow graph of the 2x2 DCT generating block.

This block is the "base" block that is repeatedly used in the PROCESS phase of all the NxN DCTs, where N is a power of 2.

In particular:

the flow graph for a 2x2 DCT is shown in Fig. 2, wherein $A = B = C = 1$ and the input and output data are pixels in the positions $(0,0)$, $(0,1)$, $(1,0)$, $(1,1)$;

for sixteen 2x2 DCTs, the inputs and the outputs are eight-component vectors and the following symbols are used, considering $A = B = C = I_{8 \times 8}$;

for four 4x4 DCTs the inputs and outputs are eight-component vectors and the following symbols are used:

$$A = \begin{cases} 2C_8^1 \times I_{8 \times 8} & \text{per stage} = 0 \\ \begin{bmatrix} (H_1)_4 & 0 \\ 0 & (H_1)_4 \end{bmatrix} & \text{per stage} = 1 \end{cases} \quad (32)$$

$$B = \begin{cases} 2C_8^3 \times I_{8 \times 8} & \text{per stage} = 0 \\ \begin{bmatrix} -(H_3)_4 & 0 \\ 0 & -(H_3)_4 \end{bmatrix} & \text{per stage} = 1 \end{cases} \quad (33)$$

$$5 \quad C = \begin{cases} 2C_4^1 \times I_{8 \times 8} & \text{per stage} = 0 \\ \begin{bmatrix} (H_2)_4 & 0 \\ 0 & (H_2)_4 \end{bmatrix} & \text{per stage} = 1 \end{cases} \quad (34)$$

for an 8x8 DCT, the inputs and outputs are eight-component vectors and the following symbols are used:

$$10 \quad A = \begin{cases} 2C_8^1 \times I_{8 \times 8} & \text{per stage} = 0 \\ (H_2)_8 & \text{per stage} = 1 \end{cases} \quad (35)$$

$$15 \quad B = \begin{cases} 2C_8^3 \times I_{8 \times 8} & \text{per stage} = 0 \\ -(H_6)_8 & \text{per stage} = 1 \end{cases} \quad (36)$$

$$C = \begin{cases} 2C_4^1 \times I_{8 \times 8} & \text{per stage} = 0 \\ (H_4)_8 & \text{per stage} = 1 \end{cases} \quad (37)$$

In the scaleable architecture for calculating an 8x8 DCT or four 4x4 DCTs (in parallel) or sixteen 2x2 DCTs (in parallel), the inputs and the outputs are vectors of eight components and the following symbols are used:

$$25 \quad A = \begin{cases} 2C_8^1 \times I_{8 \times 8} & \text{per (stage, size) = (0,0) or (0,1)} \\ I_{8 \times 8} & \text{per (stage, size) = (0,2) or (1,2)} \\ (H_2)_8 & \text{per (stage, size) (1,0)} \\ \begin{bmatrix} (H_1)_4 & 0 \\ 0 & (H_1)_4 \end{bmatrix} & \text{per (stage, size) (1,1)} \end{cases} \quad (38)$$

$$B = \begin{cases} 2C_8^3 \times I_{8 \times 8} & \text{per (stage, size) = (0,0) or (0,1)} \\ I_{8 \times 8} & \text{per (stage, size) = (0,2) or (1,2)} \\ -(H_6)_8 & \text{per (stage, size) = (1,0)} \\ \begin{bmatrix} -(H_3)_4 & 0 \\ 0 & -(H_3)_4 \end{bmatrix} & \text{per (stage, size) = (1,1)} \end{cases} \quad (39)$$

$$C = \begin{cases} 2C_4^1 \times I_{8 \times 8} & \text{per (stage, size) = (0,0) or (0,1)} \\ I_{8 \times 8} & \text{per (stage, size) = (0,2) or (1,2)} \\ (H_4)_8 & \text{per (stage, size) = (1,0)} \\ \begin{bmatrix} (H_2)_4 & 0 \\ 0 & (H_2)_4 \end{bmatrix} & \text{per (stage, size) = (1,1)} \end{cases} \quad (40)$$

Figure 3 illustrates the architecture for calculating sixteen 2x2 DCTs in parallel.

The pixels that constitute the input block are ordered during the INPUT phase and processed during the PROCESS phase to obtain the coefficients of the sixteen 2-D DCTs on four samples. For example, the 2-D DCT of the block (0,1) constituted by

$\{1[0], m[0], n[0], o[0]\}$ is $\{a[0], b[0], c[0], d[0]\}$.

The coefficients of the 2-D DCTs are rearranged during the ORDER phase in eight vectors of eight components. For example the coefficients $\{a[0], b[0], c[0], d[0]\}$ will constitute the vector 1'.

The sixteen two-component vectors so obtained are sent to the PROCESS phase to obtain the coefficients of the 2x2 DCT. These coefficients, reordered during the OUTPUT phase, constitute the output block.

Figure 4 shows the ordering of the input data for calculating sixteen 2x2 DCTs.

This figure shows the way the pixels of the 8x8 input

block are ordered to constitute the vectors of 8 components l, m, \dots, s . In each quadrant (i, j) , with $0 \leq i, j \leq 3$, the pixels belonging to the vectors are symbolized by different shadings. For example:

$$5 \quad \{A_{i,k}^{0,0}\}_{k=0}^1 = \{x_{0,0}, x_{1,1}\}$$

From each of these vectors, the components with the same index (that is the pixels with the same column index) will form a vector of four components. For example the vector l is constituted by the elements

$$10 \quad \{A1[0], B1[0]\}.$$

Therefore, each pixel of the 8×8 input block will constitute a component of one of the vectors l, m, n, o, p, q, r, s .

Figure 5 shows the process phase for calculating sixteen 2×2 DCTs.

This phase includes processing the eight-component vectors l, m, \dots, s . The PROCESS phase, which is the only phase in which arithmetical operations are performed, is executed only once to calculate in parallel the sixteen 2-D DCTs.

Figure 6 illustrates the architecture for calculating four 4×4 DCTs.

The pixels that constitute the input block are ordered in the INPUT phase and processed in the PROCESS phases to obtain the coefficients of the sixteen 1-D DCTs on 4 samples. For example, the 1-D DCT of the sequence $\{l[0], m[0], n[0], \text{ or } [0]\}$ is $\{a[0], b[0], c[0], d[0]\}$.

The coefficients of the 1-D DCTs are reordered in the ORDER phase in 8 vectors of eight components. For example the coefficients $\{a[0], b[0], c[0], d[0]\}$ will constitute the vector l' .

The 4 four-component vectors so obtained are sent to the PROCESS phase to obtain the coefficients of the 4×4

DCT. These coefficients, reordered in the OUTPUT phase, constitute the output block.

Figure 7 shows the arrangement of the input data for calculating four 4x4 DCTs.

5 This figure shows how the pixels of the 8x8 input block are ordered to constitute the eight-component vectors l, m, \dots, s .

In each quadrant (i, j) , with $0 \leq i, j \leq 3$, the pixels belonging to the different vectors have different
10 shadings. For example:

$$\{A_{i,k}^{0,0}\}_{k=0}^3 = \{x_{0,0}, x_{1,1}, x_{2,2}, x_{3,3}\}$$

From each of these vectors, the components with the same index (that is, the pixels with the same column index) will form a vector of four components. For
15 example the vector l is constituted by the elements $\{A1[0], A3[0], B3[0], B1[0]\}$.

The outcome is that each pixel of the input 8x8 block will constitute one component of one of the vectors l, m, n, o, p, q, r, s .

20 Figure 8 depicts the PROCESS phase for calculating the four 4x4 DCTs.

This phase includes processing the eight-component vectors: l, m, \dots, s .

The PROCESS phase, which is the only phase wherein
25 arithmetical operations are performed, is carried out twice:

the first time ($stage = 0$), to calculate in parallel the sixteen 1-D DCTs;

the second time ($stage = 1$), to calculate the 8x8
30 DCT starting from the coefficients of the 1-D DCTs.

$$A = \begin{cases} 2C_8^1 \times I_{8 \times 8} & \text{per stage} = 0 \\ \begin{bmatrix} (H_1)_4 & 0 \\ 0 & (H_1)_4 \end{bmatrix} & \text{per stage} = 1 \end{cases}, \quad (41)$$

$$B = \begin{cases} 2C_8^3 \times I_{8 \times 8} & \text{per stage} = 0 \\ \begin{bmatrix} -(H_3)_4 & 0 \\ 0 & -(H_3)_4 \end{bmatrix} & \text{per stage} = 1 \end{cases} \quad (42)$$

$$C = \begin{cases} 2C_4^1 \times I_{8 \times 8} & \text{per stage} = 0 \\ \begin{bmatrix} (H_2)_4 & 0 \\ 0 & (H_2)_4 \end{bmatrix} & \text{per stage} = 1 \end{cases} \quad (43)$$

$$t = \begin{cases} 1 & \text{per stage} = 0 \\ 2 & \text{per stage} = 1 \end{cases} \quad (44)$$

Figure 9 illustrates the architecture for calculating an 8x8 DCT.

The pixels that constitute the input block are ordered during the INPUT phase and are processed in the PROCESS phase to obtain the coefficients of the eight 1-D DCTs on 8 samples. For example, the 1-D DCT of the sequence $\{l[0], m[0], \dots, s[0]\}$ is $\{a[0], b[0], \dots, h[0]\}$.

The coefficients of the 1-D DCTs are rearranged during the ORDER phase in 8 vectors of eight components. For example the coefficients $\{a[0], b[0], \dots, h[7]\}$ will constitute the l' vector.

The 8 eight-component vectors so obtained are sent to the PROCESS phase to obtain the 8x8 DCT coefficients. These coefficients, rearranged during the OUTPUT phase, constitute the output block.

Figure 10 shows the arrangement of the input data for calculating an 8x8 DCT.

This figure shows how the pixels of the input 8x8 block are arranged to constitute the 8 eight-component vectors: l, m, \dots, s . The pixels belonging to the vectors $A1, A3, A5, A7, B7, B5, B3, B1$ are symbolized with different shadings, for example:

$$\{A_{i,j}\}_{i=0}^7 = \{x_{0,0}, x_{1,1}, x_{2,2}, x_{3,3}, x_{4,4}, x_{5,5}, x_{6,6}, x_{7,7}\}$$

From each of these vectors, the components with the same index (that is, the pixels with the same column index) will form a vector of eight components. For example, the vector l is constituted by the elements
 5 $\{A1[0], A3[0], \dots, B1[0]\}.$

The result is that each pixel of the input 8x8 block will constitute a component of one of the vectors $l, m, n, o, p, q, r, s.$
 10

Figure 11 depicts the PROCESS phase for calculating an 8x8 DCT.

This phase includes processing the eight-component vectors $l, m, \dots, s.$

15 The PROCESS phase in which arithmetical operations are performed is executed twice:

the first time ($stage=0$), to calculate in parallel the sixteen 1-D DCTs;

the second time ($stage=1$), to calculate the 8x8 DCT starting from the coefficients of the 1-D DCTs.

20 In Fig. 11 the following symbols have been used:

$$A = \begin{cases} 2C_8^1 \times I_{8 \times 8} & \text{per stage} = 0 \\ (H_2)_8 & \text{per stage} = 1 \end{cases} \quad (45)$$

$$25 \quad B = \begin{cases} 2C_8^3 \times I_{8 \times 8} & \text{per stage} = 0 \\ -(H_6)_8 & \text{per stage} = 1 \end{cases} \quad (46)$$

$$C = \begin{cases} 2C_4^1 \times I_{8 \times 8} & \text{per stage} = 0 \\ (H_4)_8 & \text{per stage} = 1 \end{cases} \quad (47)$$

$$30 \quad t = \begin{cases} 1 & \text{per stage} = 0 \\ 2 & \text{per stage} = 1 \end{cases} \quad (48)$$

Figure 12 illustrates a scaleable architecture for calculating an 8x8 DCT or four 4x4 DCTs or sixteen 2x2 DCTs.

The pixels that constitute the input block are ordered

during the INPUT phase and processed during the PROCESS phase, which calculates:

- the 1-D DCTs (for $stage = 0$, that is for the 8x8 DCT, and for $stage = 1$, that is for the 4x4 DCTs;
- 5 the 2-D DCTs for $stage = 2$ directly, that is for the 2x2 DCTs;

When $stage = 0$ and $stage = 1$ the coefficients are then rearranged in the ORDER phase in 8 eight-component vectors, which are sent to the PROCESS phase to obtain
 10 the coefficients of the 2-D DCT. These coefficients, rearranged in the OUTPUT phase, constitute the output block.

If $stage = 2$ the coefficients are transmitted directly to the OUTPUT phase, where they are rearranged
 15 to constitute the output block.

Figure 13 depicts the INPUT phase for a scaleable architecture.

The inputs are the 64 pixels that constitute the input block.

20 The arrangement of the inputs is operated through the MUXes controlled by the *size* variable.

The 64 outputs are the 8 vectors of eight components $1, m, \dots, s$.

Figure 14 depicts the PROCESS phase for a scaleable
 25 architecture.

This phase includes calculating in parallel the eight 1-D DCTs by processing the vectors $1, m, \dots, s$ as shown in Fig. 11.

In this figure we may notice that the use of 16 MUXes
 30 controlled by *size*.

The eight MUXes on the left serve to bypass the necessary operations only for calculating the 8x8 DCT; therefore, the bypass takes place for $stage = 1$ or 2, while it does not occur when $stage = 0$.

The eight MUXes on the right serve to output only the result corresponding to the pre-selected size.

In Fig. 14 the following symbols are used:

$$t = \begin{cases} 1 & \text{per stage} = 0 \\ 2 & \text{per stage} = 1 \end{cases} \quad (49)$$

$$A = \begin{cases} 2C_8^1 \times I_{8 \times 8} & \text{per (stage, size) = (0,0) or (0,1)} \\ I_{8 \times 8} & \text{per (stage, size) = (0,2) or (1,2)} \\ (H_2)_8 & \text{per (stage, size) (1,0)} \\ \begin{bmatrix} (H_1)_4 & 0 \\ 0 & (H_1)_4 \end{bmatrix} & \text{per (stage, size) (1,1)} \end{cases} \quad (50)$$

$$= \begin{cases} 2C_8^3 \times I_{8 \times 8} & \text{per (stage, size) = (0,0) or (0,1)} \\ I_{8 \times 8} & \text{per (stage, size) = (0,2) or (1,2)} \\ -(H_6)_8 & \text{per (stage, size) (1,0)} \\ \begin{bmatrix} -(H_3)_4 & 0 \\ 0 & -(H_3)_4 \end{bmatrix} & \text{per (stage, size) (1,1)} \end{cases} \quad (51)$$

$$' = \begin{cases} 2C_4^1 \times I_{8 \times 8} & \text{per (stage, size) = (0,0) or (0,1)} \\ I_{8 \times 8} & \text{per (stage, size) = (0,2) or (1,2)} \\ (H_4)_8 & \text{per (stage, size) (1,0)} \\ \begin{bmatrix} (H_2)_4 & 0 \\ 0 & (H_2)_4 \end{bmatrix} & \text{per (stage, size) (1,1)} \end{cases} \quad (52)$$

$$D = \begin{cases} 2C_{16}^1 \times I_{8 \times 8} & \text{per stage} = 0 \\ (H_1)_8 & \text{per stage} = 1 \end{cases} \quad (53)$$

$$E = \begin{cases} 2C_{16}^3 \times I_{8 \times 8} & \text{per stage} = 0 \\ (H_5)_8 & \text{per stage} = 1 \end{cases} \quad (54)$$

$$F = \begin{cases} 2C_{16}^7 \times I_{8 \times 8} & \text{per stage} = 0 \\ -(H_7)_8 & \text{per stage} = 1 \end{cases} \quad (55)$$

$$G = \begin{cases} 2C_{16}^5 \times I_{8 \times 8} & \text{per stage} = 0 \\ -(H_3)_8 & \text{per stage} = 1 \end{cases} \quad (56)$$

Figure 15 is a block diagram of the structure that implements the PROCESS phase.

5 For example, the QA block receives as an input two vectors of eight components (each component is a pixel, that may have already been processed) and outputs two vectors of eight components. The first vector is the sum of the two input vectors, while the second vector is the difference between the two input vectors, successively
10 processed with the linear operator A. It should be noticed that the A, B, C, D, E, F, G operators are 8x8 matrices.

Figure 16 is a detailed scheme of the QA block.

15 This scheme shows the details of the single components of the two input vectors and the arithmetical operators (adders etc.) which act on each component. The results are sent to the MUXes depicted on the right side of the figure, each of which, depending on the control
20 variables *stage* and *size*, select only one result, which constitutes one component of the output vector.

Figure 17 is a detailed scheme of the QB block.

This scheme shows the details of the single components of the two input vectors and the arithmetical
25 operators (adders etc.) which act on each component. The results are sent to the MUXes depicted on the right side of the figure, each of which, depending on the control variables *stage* and *size*, select only one result, which constitute a component of the output vector.

30 Figure 18 is a detailed scheme of the QC block.

This scheme shows the details of the single components of the two input vectors and the arithmetical

operators (adders etc.) acting on each component. The results are sent to the MUX_{es} depicted on the right side of the figure, each of which, depending on the control variable *stage* and *size*, select only one result, which

5 constitute one component of the output vector.

Figure 19 is a detailed scheme of the QD block.

This scheme shows the details of the single components of the two input vectors and the arithmetical operators (adders etc.) which act on each component. The

10 results are sent to the MUX_{es} depicted on the right side of the figure, each of which, depending on the control variable *stage* and *size*, select only one result, which constitute a component of the output vector.

Figure 20 is a detailed scheme of the QE block.

15 This scheme shows the details of the single components of the two input vectors and the arithmetical operators (adders etc.) which act on each component. The results are sent to the MUX_{es} depicted on the right side of the figure, each of which, depending on the control

20 variable *stage*, select only one result, which constitute one component of the output vector.

Figure 21 is a detailed scheme of the QF block.

This scheme shows the details of the single components of the two input vectors and the arithmetical

25 operators (adders etc.) which act on each component. The results are sent to the MUX_{es} depicted on the right side of the figure, each of which, depending on the control variable {*stage*} select only one result, which constitute a component of the output vector.

30 Figure 22 is a detailed scheme of the QG block.

This scheme shows the details of the single components of the two input vectors and the arithmetical operators (adders etc.) which act on each component. The results are sent to the MUX_{es} depicted on the right side

35 of the figure, each of which, depending on the control

variable *stage*, select only one result, which constitute a component of the output vector.

Figure 23 depicts the ORDER phase for the scaleable architecture.

5 The inputs are constituted by the 64 pixels after they have been processed through the PROCESS phase.

 The inputs arrangement is effected by the MUXes controlled by the variable *size*.

10 The 64 outputs are the components of the eight-component vectors *l*, *m*, ..., *s*.

Figure 24 depicts the OUTPUT phase for the scaleable architecture.

15 The inputs are constituted by the 64 2-D DCT coefficients. The input arrangement is effected by the MUXes controlled by the variable *size*.

 The 64 outputs are the pixels that constitute the output block.

662000 "12300000

THAT WHICH IS CLAIMED IS

1. A method of calculating the discrete cosine transform (DCT) of blocks of pixels of a picture, characterized in that it comprises the steps of
 5 defining first subdivision blocks called *range* blocks, having a fractional and scaleable size $N/2^i \times N/2^i$, where i is an integer number, in respect to a maximum pre-defined size of $N \times N$ pixels of blocks of division of said picture, referred to as *domain* blocks, shiftable
 10 by intervals of $N/2^i$ pixels, and of calculating the DCT on 2^i *range* blocks of subdivision of a *domain* block of $N \times N$ pixels of said picture, in parallel.

2. The method according to claim 1, characterized in
 15 that the calculation of the DCT in parallel on all *range* blocks of subdivision of a certain *domain* block is carried out in a hardware structure and comprises the steps of:

- a) ordering the pixels in function of a subdivision in
 20 *range* blocks of a certain dimension by rearranging the input pixels in a number 2^i of sequences or vectors of 2^i components;
- b) calculating in parallel 2^i monodimensional DCTs by processing said vectors defined in the preceding step
 25 a);
- c) arranging the output sequences of the monodimensional DCTs relative to said 2^i vectors;
- d) completing the calculation in parallel of 2^i
 30 bidimensional DCTs by processing said output sequences of monodimensional DCTs produced in step c);
- e) arranging the output sequences of bidimensional DCTs generated in step d) in a number 2^i of vectors of bidimensional DCT coefficients.

3. The method according to claim 2, characterized in that the calculation in parallel of said 2^i monodimensional DCTs in step b) and the completion of
 5 the parallel calculation of 2^i bidimensional DCTs of step d) are performed by subdividing the sequences resulting from step a) and from step c), respectively, in groups of scalar elements, calculating the sums and differences thereof by way of adders and subtractors and by
 10 reiterately multiplying the sum and difference results by respective coefficients until completing the calculation of the relative DCT coefficients, respectively monodimensional and bidimensional.

15 4. A method of compressing data of a picture to be stored or transmitted through a fractal coding, characterized in that the fractal transform is carried out in the domain of the discrete cosine transform (DCT) through the following steps:

20 subdividing a picture in blocks of pixels of said two distinct type of blocks as defined in claim 1;
 parallely calculating the discrete cosine transform (DCT) of all the 2^i range blocks and of a relative domain block;

25 classifying the transformed range blocks according to their relative complexity represented by the sum of the values of the three AC coefficients;

applying the fractal transform in the DCT domain to the data of the range blocks whose complexity
 30 classification exceeds a pre-defined threshold and storing only the DC coefficient of the range blocks with a complexity lower than said threshold,
 identifying a relative domain block to which the range block in a transformation belongs that produces the
 35 beset fractal approximation of the range block;

quantizing said difference picture in the DCT domain by using a quantization table preestablished in function of the characteristics of human sight;

storing or transmitting the coding code of each range block compressed in the DCT domain and the DC coefficient of each uncompressed range block.

METHOD AND SCALABLE ARCHITECTURE FOR PARALLEL
CALCULATION OF THE DCT OF BLOCKS OF PIXELS OF DIFFERENT
SIZES AND COMPRESSION THROUGH FRACTAL CODING

Abstract of the Disclosure

A method of calculating the discrete cosine transform (DCT) of blocks of pixels of a picture includes the steps of defining first subdivision blocks called range
5 blocks, having a fractional and scaleable size $N/2^i * N/2^i$, where i is an integer number, with respect to a maximum pre-defined size of $N*N$ pixels of blocks of division of the picture, referred to as domain blocks, shiftable by
10 intervals of $N/2^i$ pixels. The method also includes the step of calculating the DCT on 2^i range blocks of a subdivision of a domain block of $N*N$ pixels of the picture, in parallel.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:)
PAU ET AL.)
)
Serial No. Not Yet Assigned)
)
Filing Date: Herewith)
)
For: METHOD AND SCALABLE ARCHITECTURE))
FOR PARALLEL CALCULATION OF THE)
DCT OF BLOCKS OF PIXELS OF)
DIFFERENT SIZES AND COMPRESSION)
THROUGH FRACTAL CODING)
)

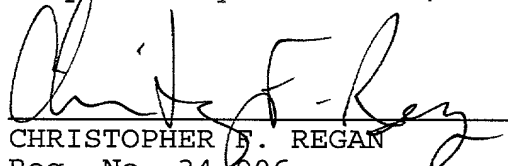
SUBMISSION OF PROPOSED MODIFICATIONS TO DRAWINGS

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

Submitted herewith is a request for a proposed drawing modification to correct an informality in FIGs. 5-7, 9, 10, 15 and 23 as indicated in red ink.

Respectfully submitted,


CHRISTOPHER F. REGAN
Reg. No. 34,906
Allen, Dyer, Doppelt, Milbrath
& Gilchrist, P.A.
255 S. Orange Ave., Suite 1401
P. O. Box 3791
Orlando, Florida 32802
(407) 841-2330
Attorney for Applicants

CERTIFICATE OF MAILING BY "EXPRESS MAIL"

"EXPRESS MAIL" MAILING LABEL NUMBER EL4364163465US

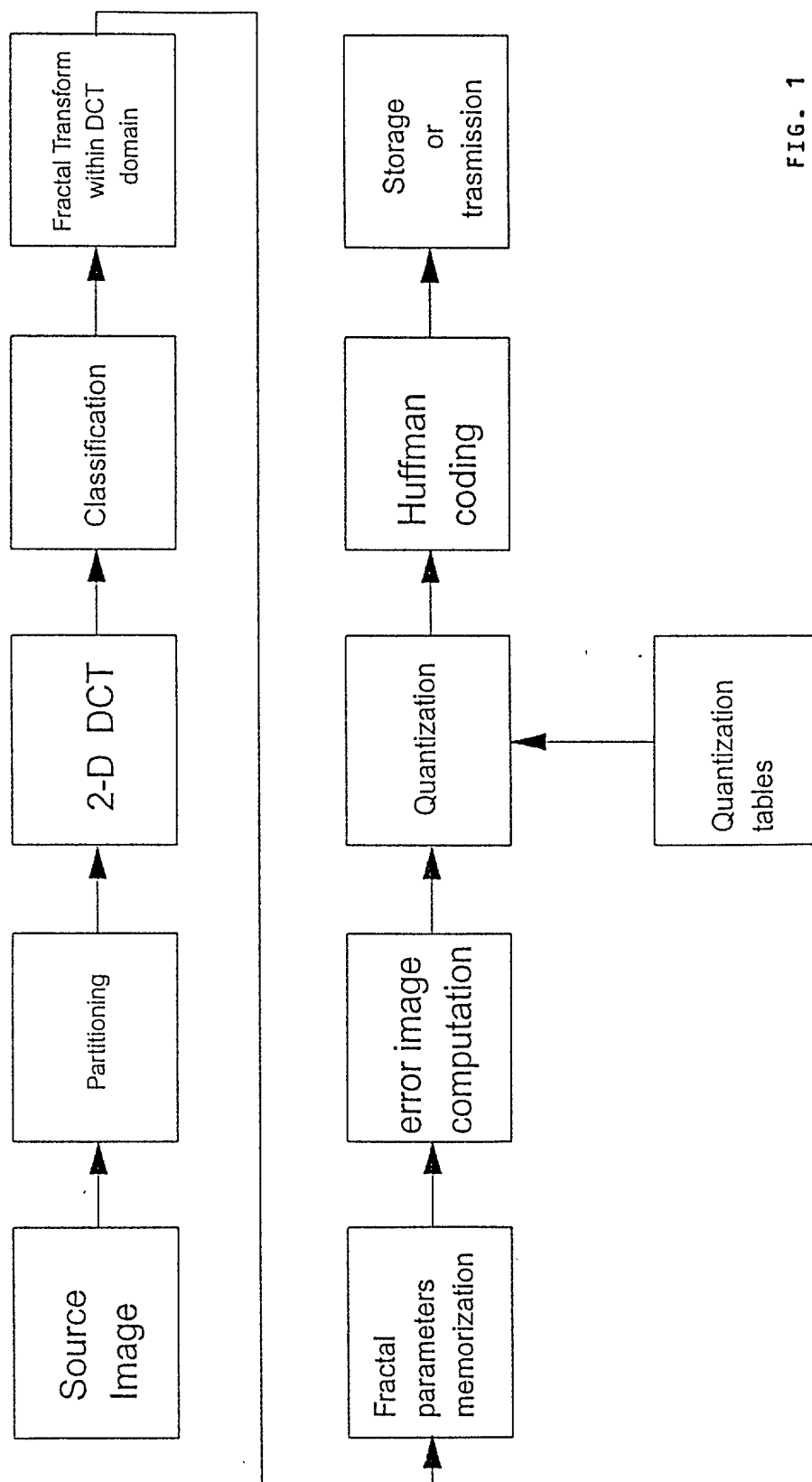
DATE OF DEPOSIT 9/3/99

I HEREBY CERTIFY THAT THIS PAPER OR FEE IS BEING DEPOSITED WITH THE U.S. POSTAL SERVICE "EXPRESS MAIL" POST OFFICE ADDRESS AT SERVICE UNDER 37 CFR 1.10 ON THE DATE INDICATED ABOVE AND IS RELAYED TO THE COMMISSIONER OF PATENTS AND TRADEMARKS, Washington, D.C. 20231

Eric Link

PRINTED NAME OF PERSON MAILING PAPER OR FEE

(SIGNATURE OF PERSON MAILING PAPER OR FEE)



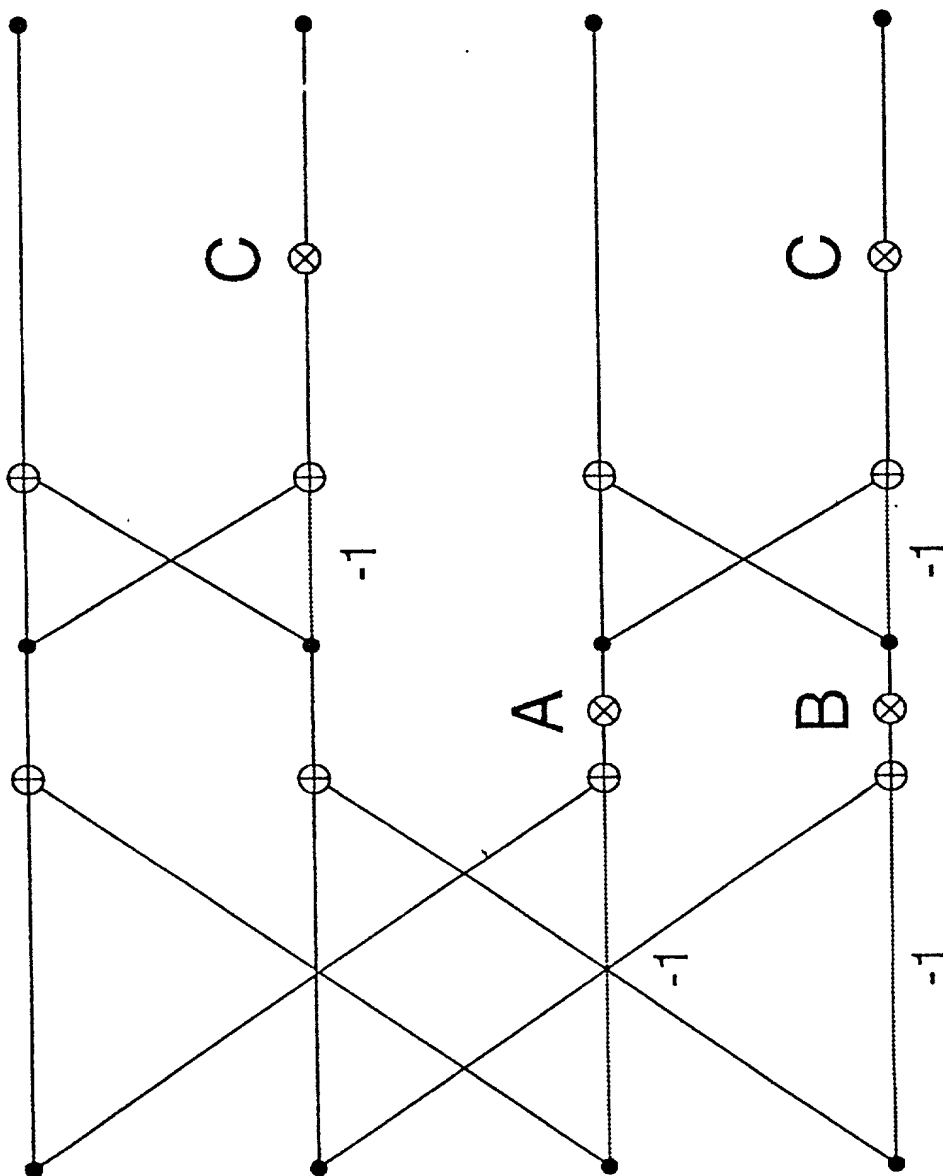


FIG. 2

66000"45500000

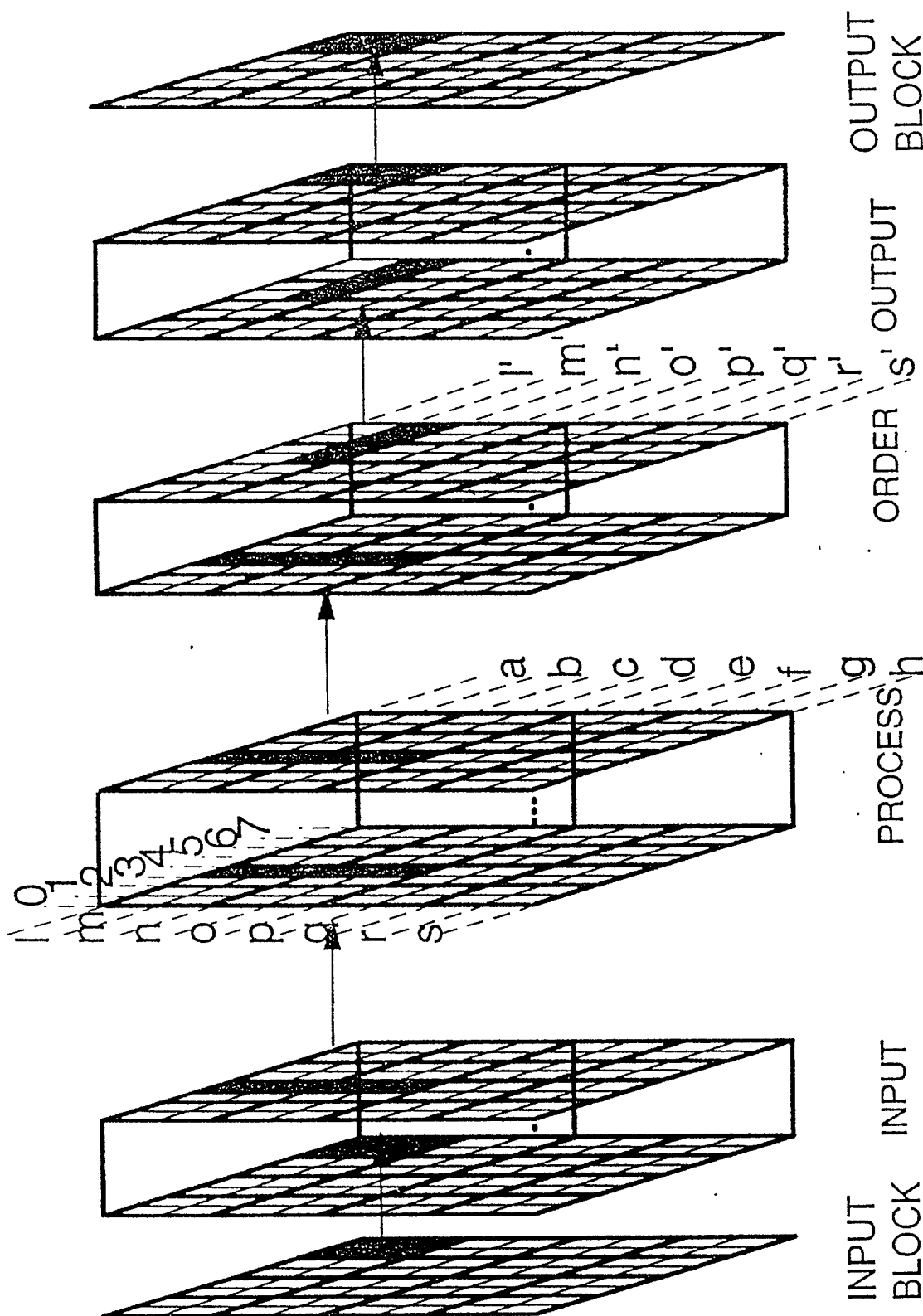


FIG. 3

6666666666666666

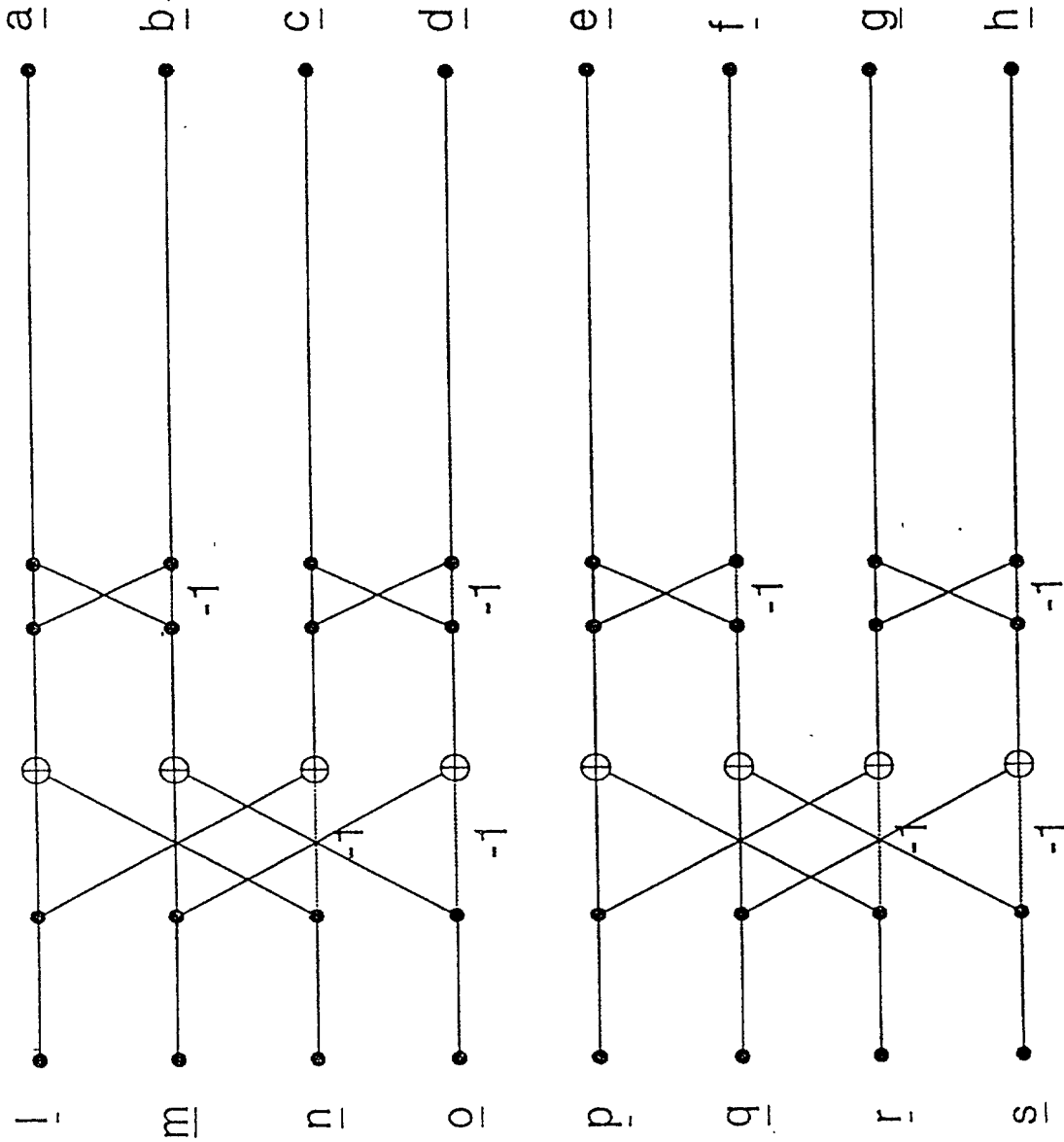


FIG. 5

“caca” 432560

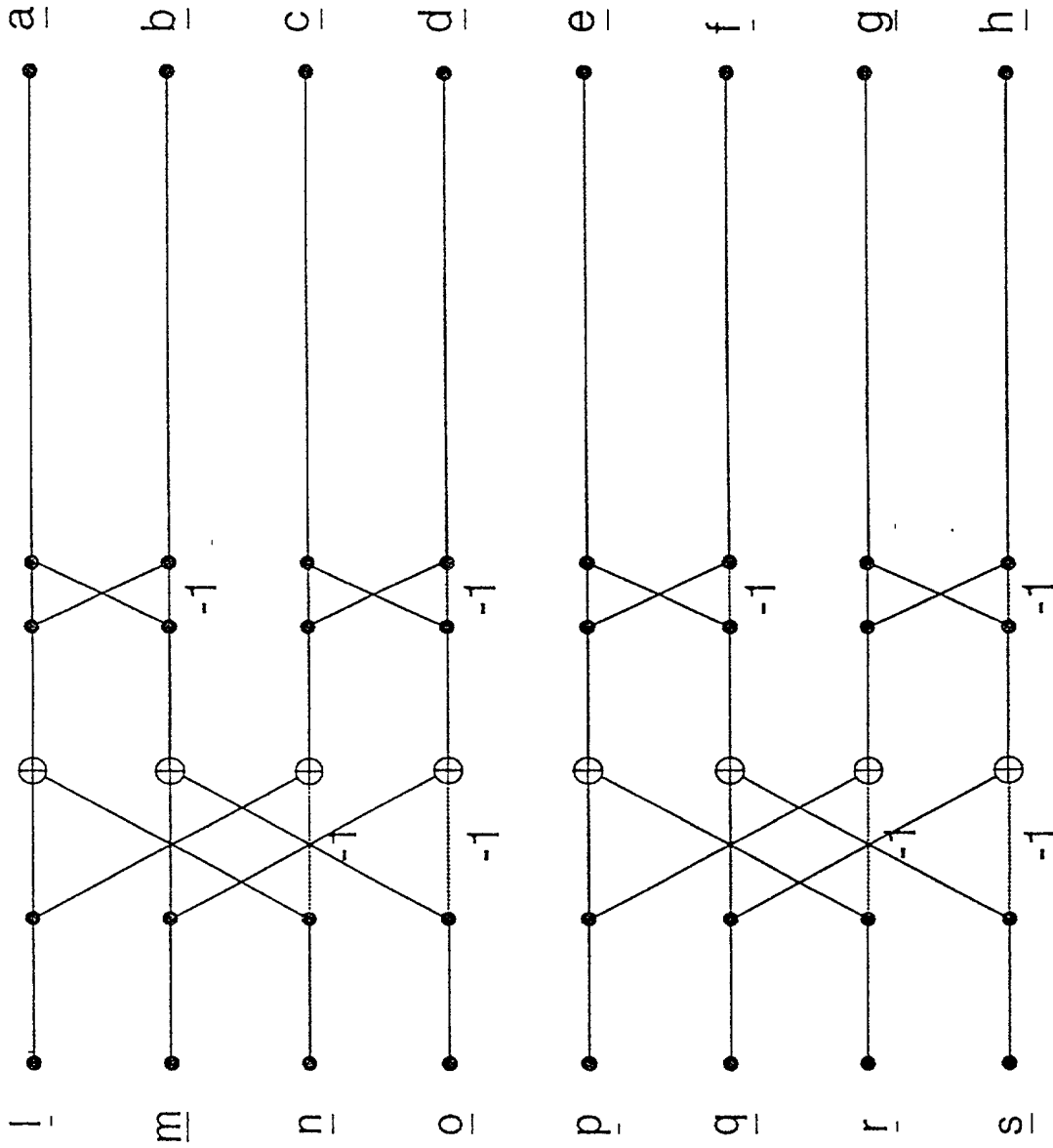


FIG. 5

FIG. 5

6/24

66000" 4500000

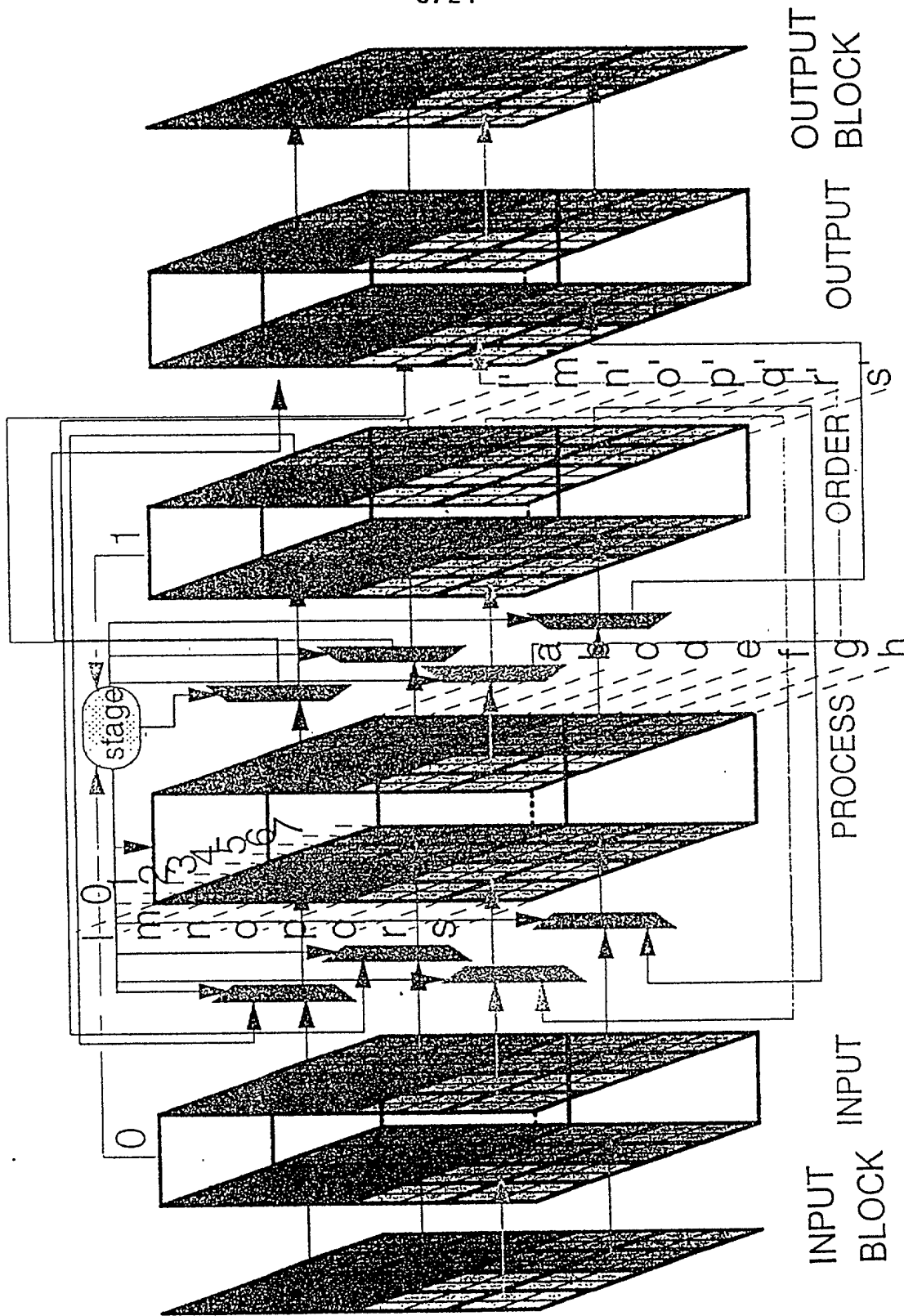
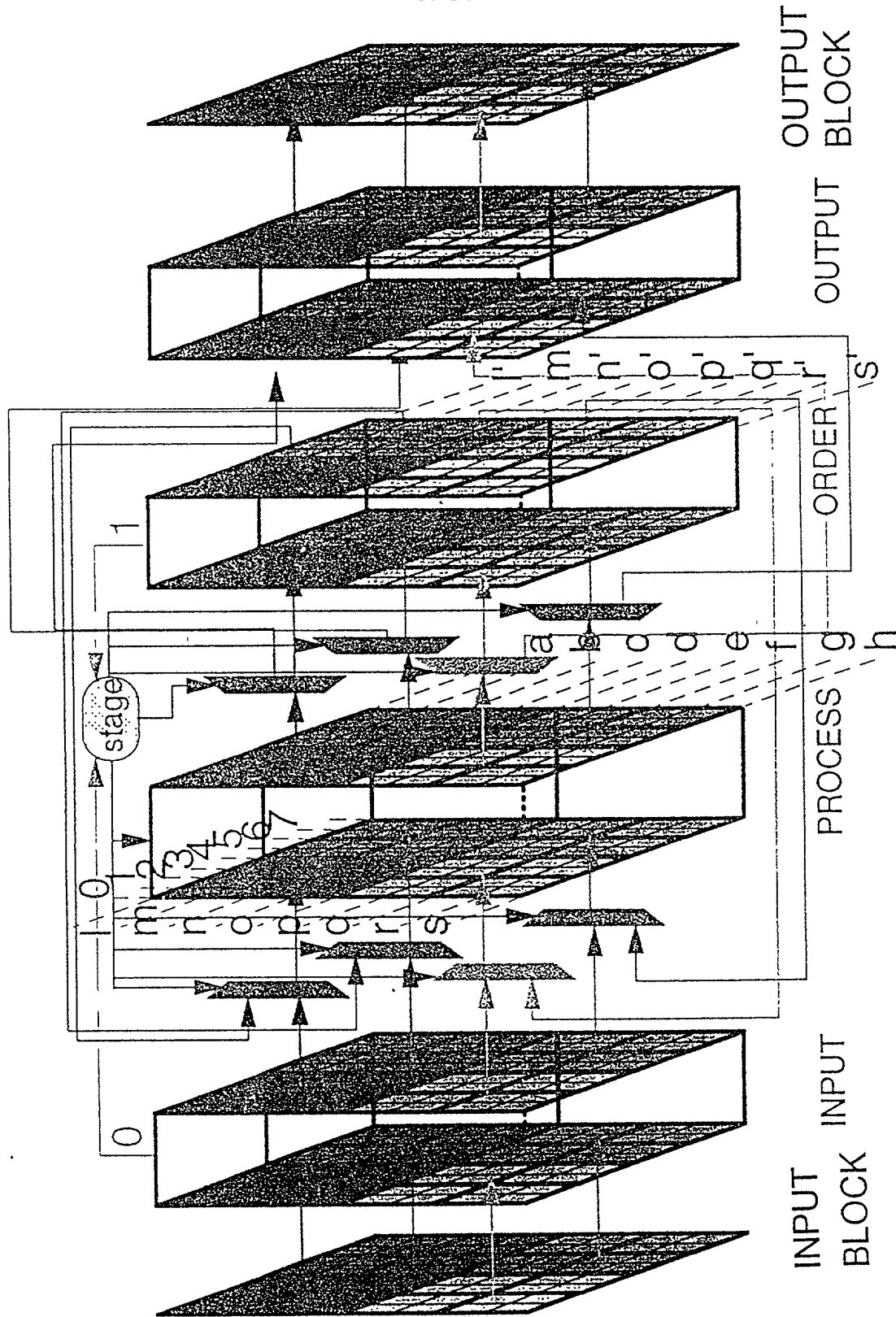


FIG. 6

66000"4550000

6/24



OUTPUT
BLOCK

OUTPUT
BLOCK

ORDER

PROCESS

INPUT
BLOCK

Fig. 6

FIG. 6

cccccccc

PIXEL								
	0	1	2	3	4	5	6	7
0	l(0)	m(2)	n(1)	o(3)	l(4)	m(6)	n(5)	o(7)
1	l(1)	m(0)	n(3)	o(2)	l(5)	m(4)	n(7)	o(6)
2	l(2)	m(3)	n(0)	o(1)	l(6)	m(7)	n(4)	o(5)
3	l(3)	m(1)	n(2)	o(0)	l(7)	m(5)	n(6)	o(4)
4	p(0)	q(2)	r(1)	s(3)	p(4)	q(6)	r(5)	s(7)
5	p(1)	q(0)	r(3)	s(2)	p(5)	q(4)	r(7)	s(6)
6	p(2)	q(3)	r(0)	s(1)	p(6)	q(7)	r(4)	s(5)
7	p(3)	q(1)	r(2)	s(0)	p(7)	q(5)	r(6)	s(4)
								QUADRANT
								0 1

FIG. 7

QUADRANT

Fig. 7

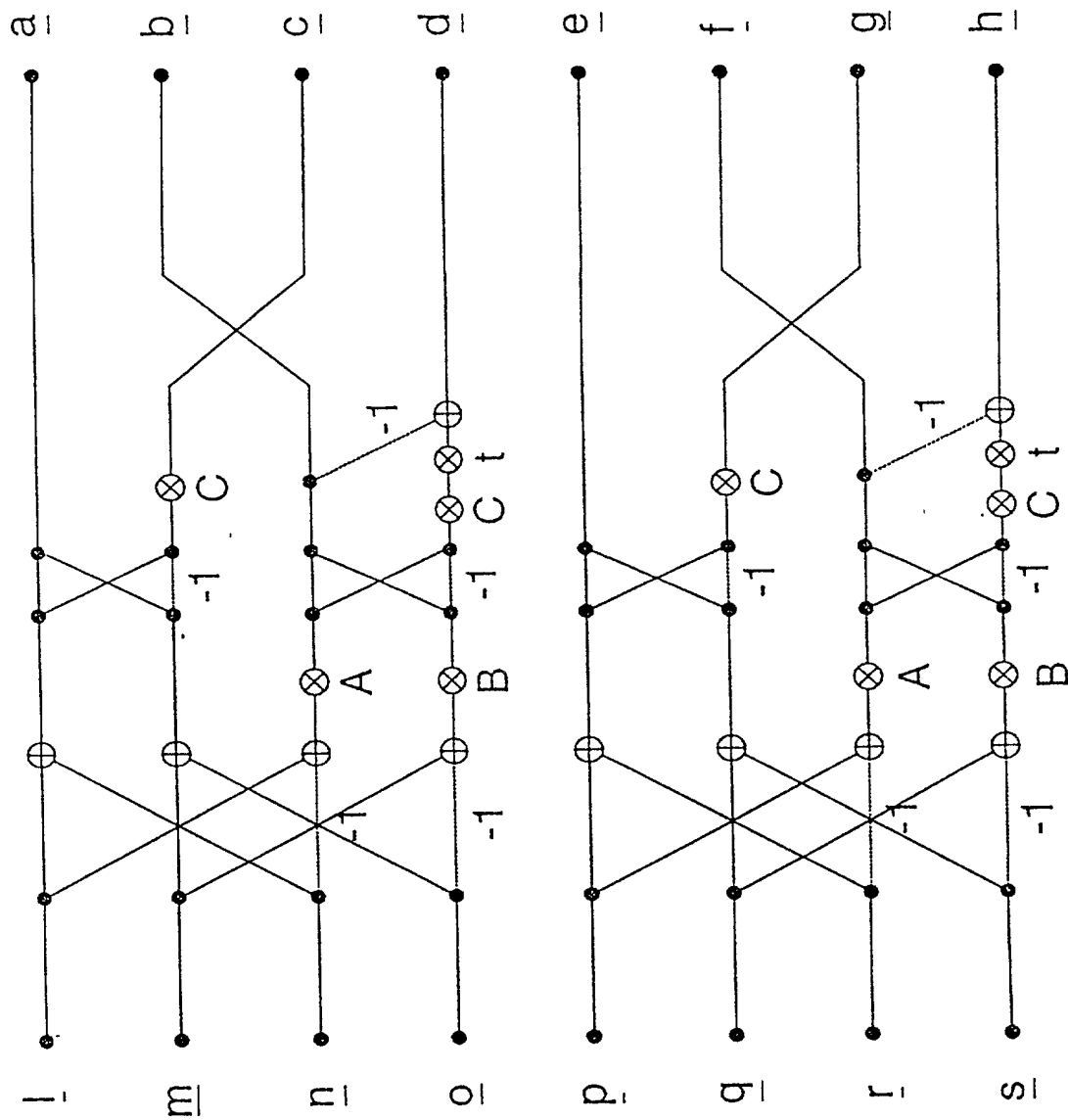


FIG. 8

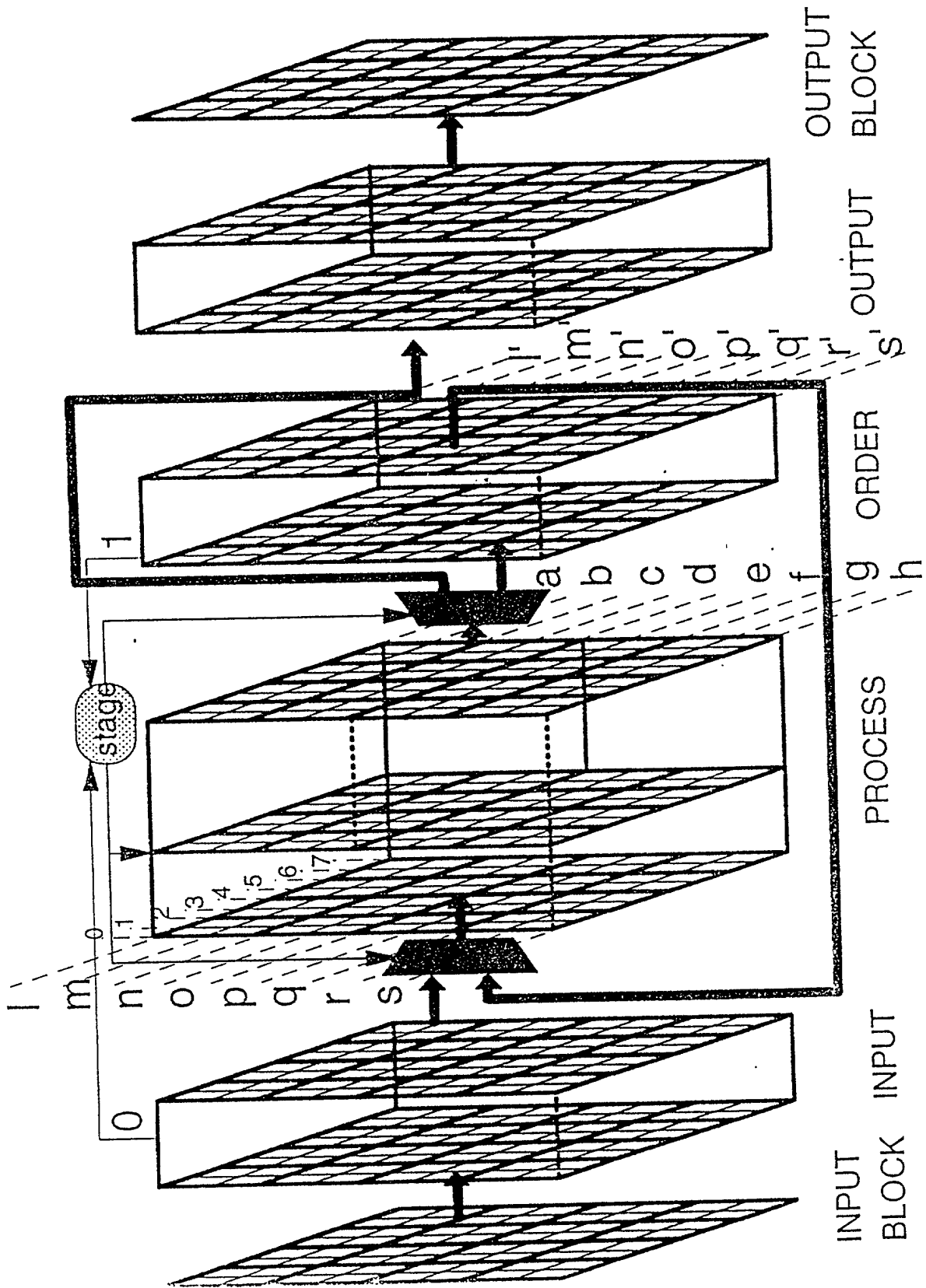


FIG. 9

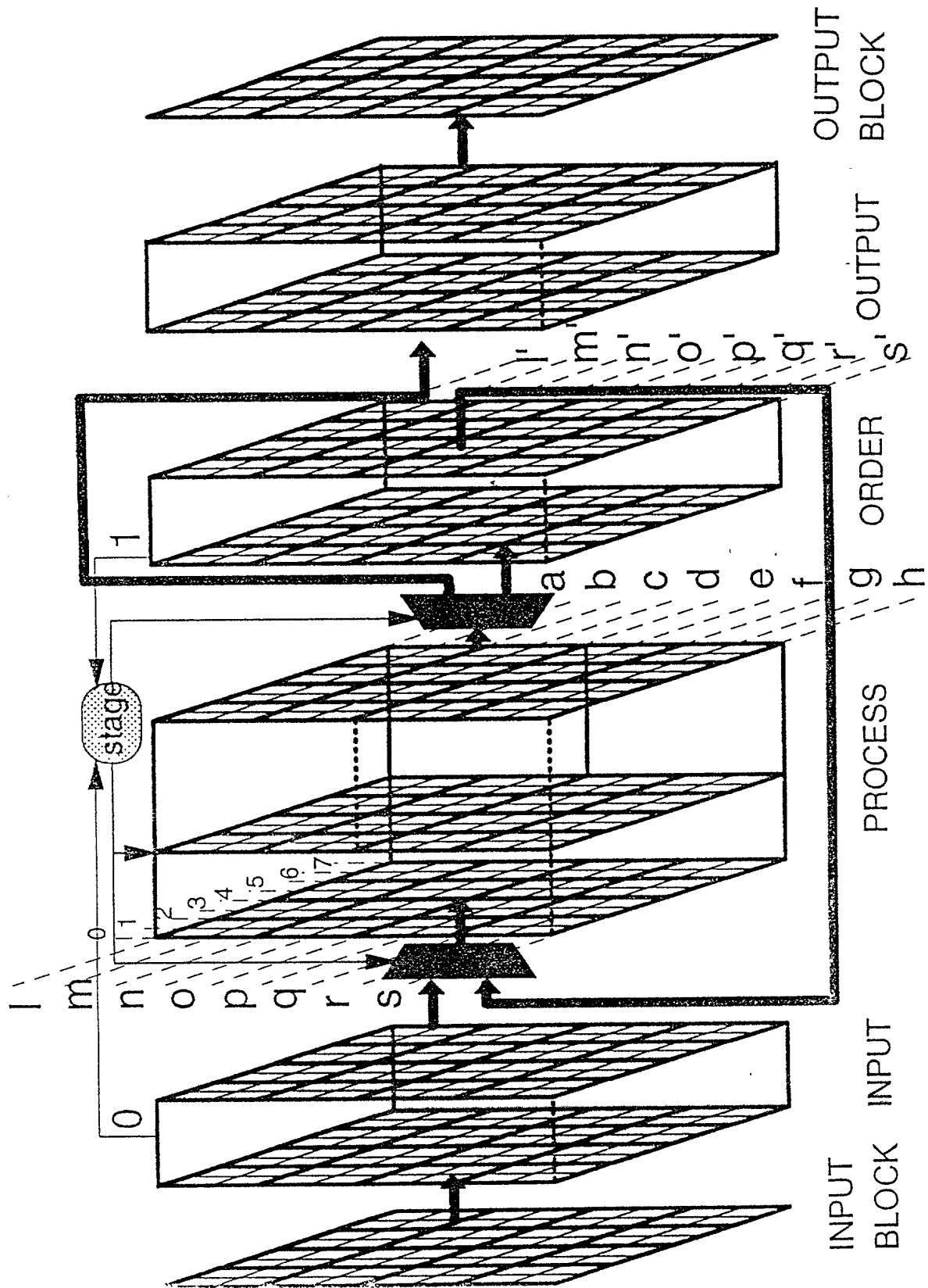


Fig. 9

0	1	2	3	4	5	6	7
l(0)	m(5)	n(6)	o(4)	p(3)	q(1)	r(2)	s(7)
l(1)	m(0)	n(3)	o(2)	p(5)	q(4)	r(7)	s(6)
l(2)	m(4)	n(0)	o(6)	p(1)	q(7)	r(3)	s(5)
l(3)	m(6)	n(2)	o(0)	p(7)	q(5)	r(1)	s(4)
l(4)	m(1)	n(5)		p(0)	q(2)	r(6)	s(3)
l(5)	m(3)		o(1)	p(6)	q(0)	r(4)	s(2)
l(6)	m(7)	n(4)	o(5)	p(2)	q(3)	r(0)	s(1)
l(7)	m(2)	n(1)	o(3)	p(4)	q(6)	r(5)	s(0)

FIG. 10

0	1	2	3	4	5	6	7
l(0)	m(5)	n(6)	o(4)	p(3)	q(1)	r(2)	s(7)
l(1)	m(0)	n(3)	o(2)	p(5)	q(4)	r(7)	s(6)
l(2)	m(4)	n(0)	o(6)	p(1)	q(7)	r(3)	s(5)
l(3)	m(6)	n(2)	o(0)	p(7)	q(5)	r(1)	s(4)
l(4)	m(1)	n(5)		p(0)	q(2)	r(6)	s(3)
l(5)	m(3)	n(7)	o(1)	p(6)	q(0)	r(4)	s(2)
l(6)	m(7)	n(4)	o(5)	p(2)	q(3)	r(0)	s(1)
l(7)	m(2)	n(1)	o(3)	p(4)	q(6)	r(5)	s(0)

FIG. 10

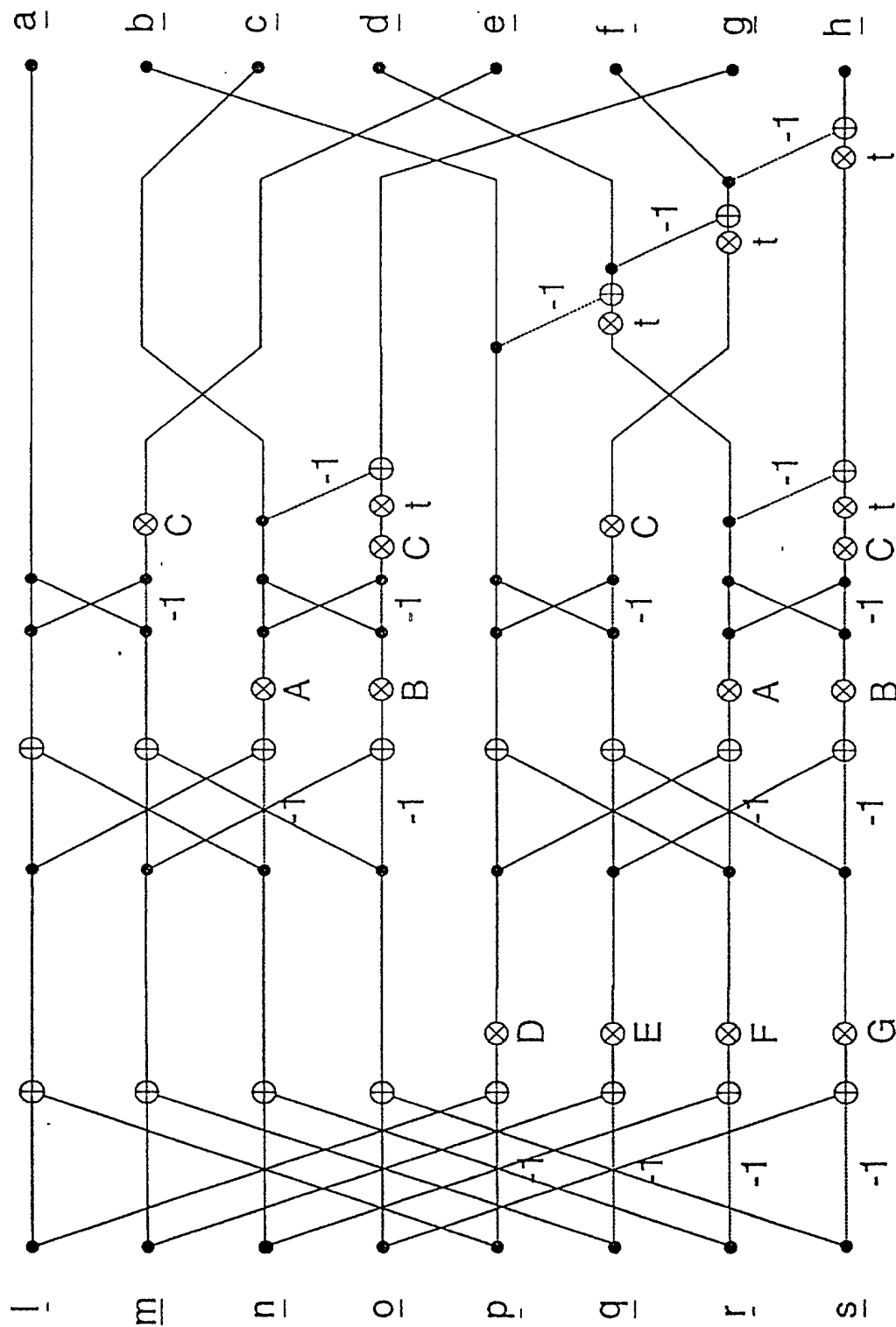


FIG. 11

66000-1556660

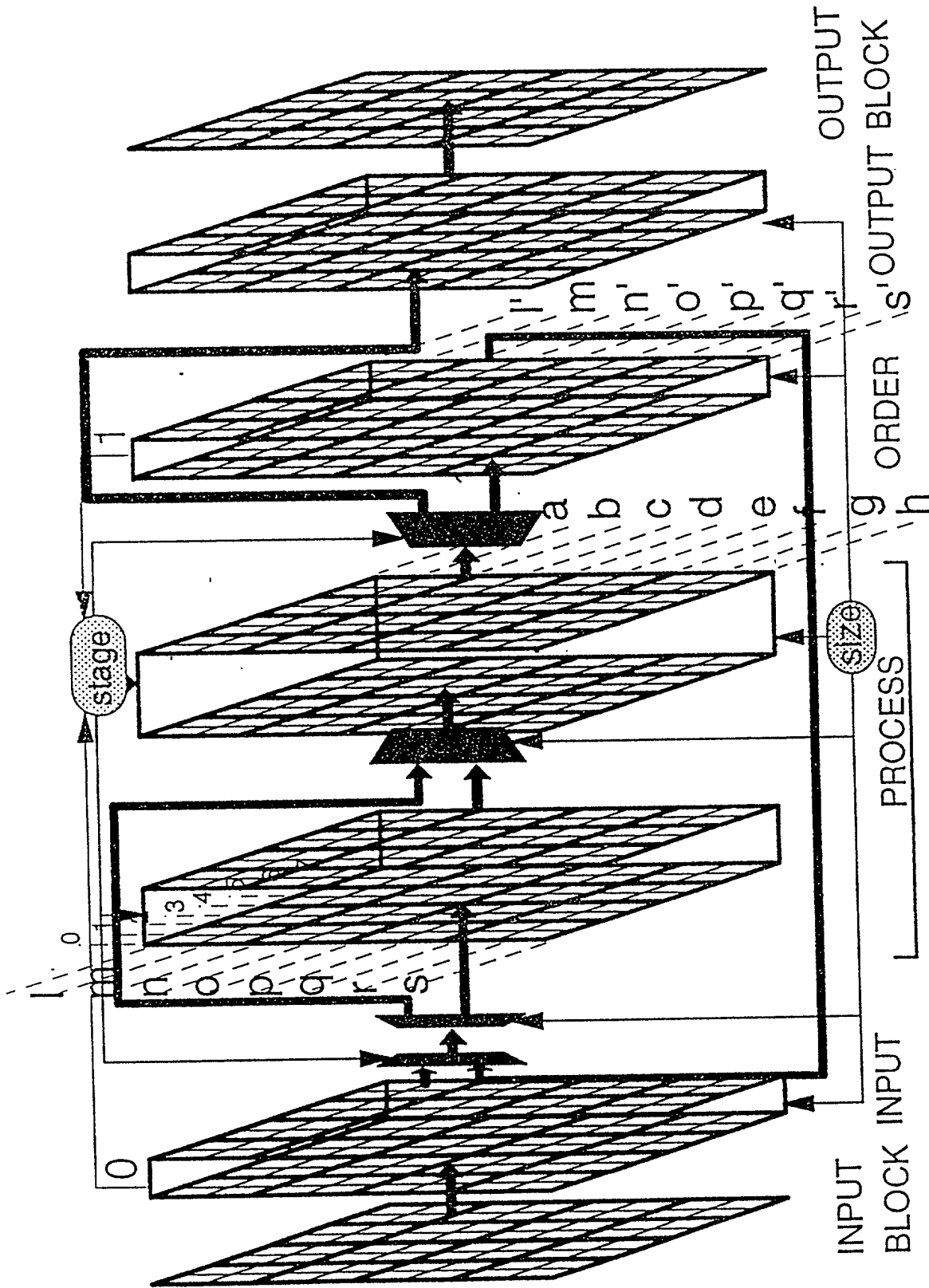


FIG. 12

66000-1530550

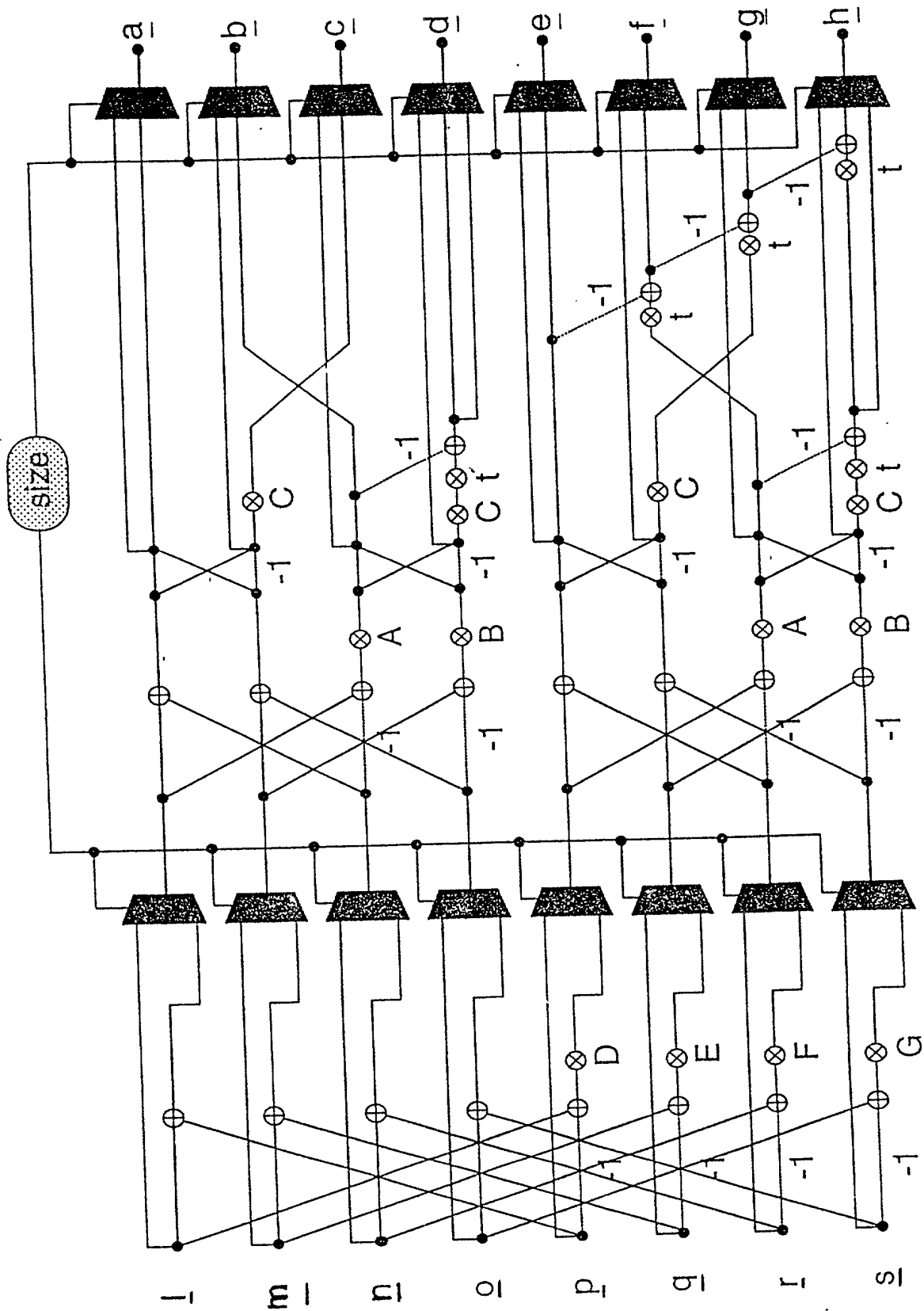
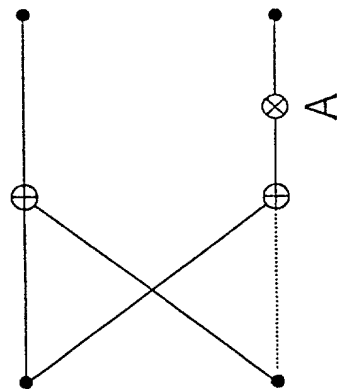
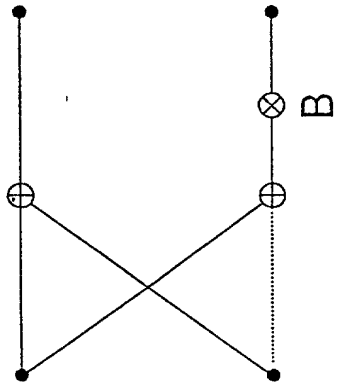


FIG. 14

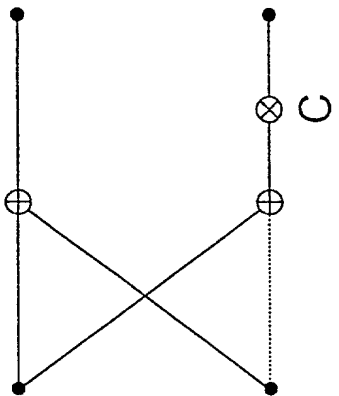
Block QA



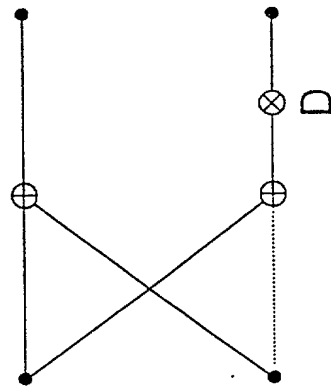
Block QB



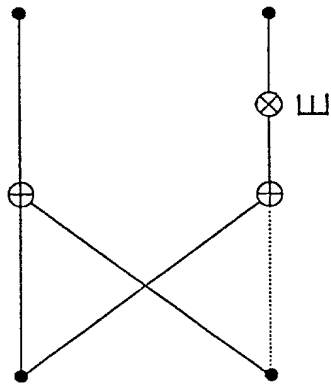
Block QC



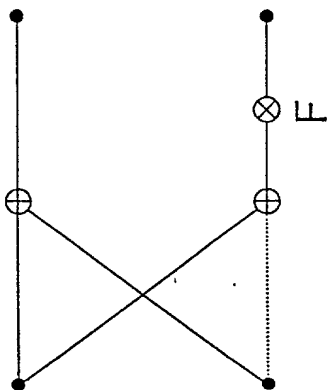
Block QD



Block QE



Block QF



Block QG

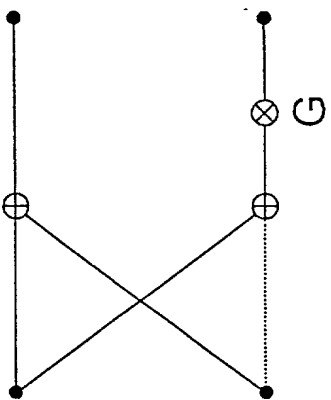
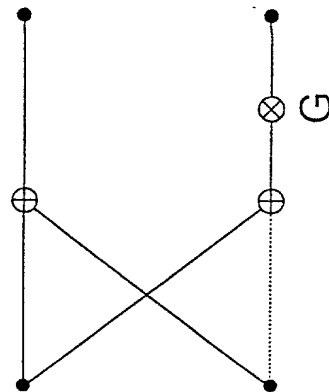
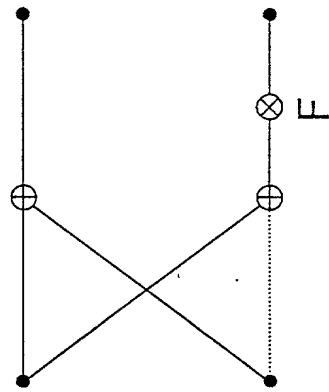
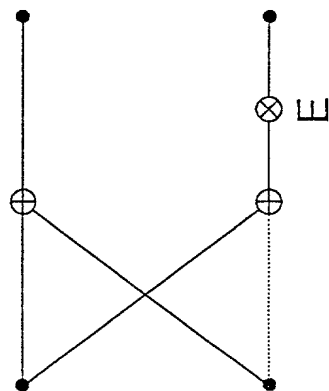
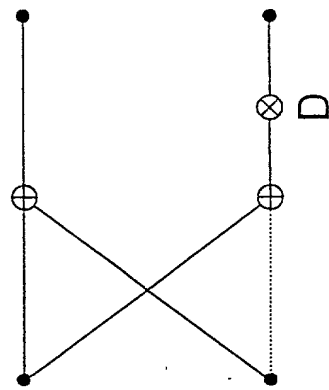
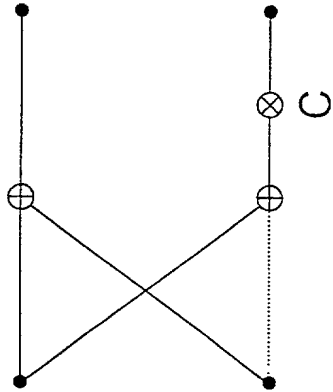
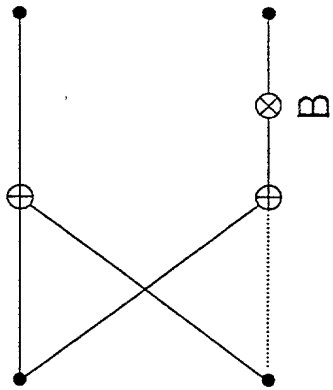
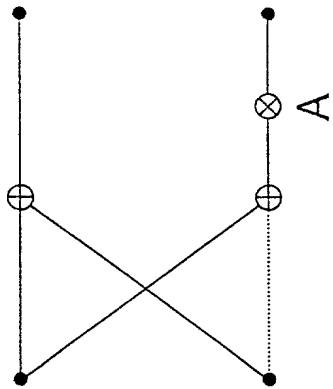


FIG. 15



5191

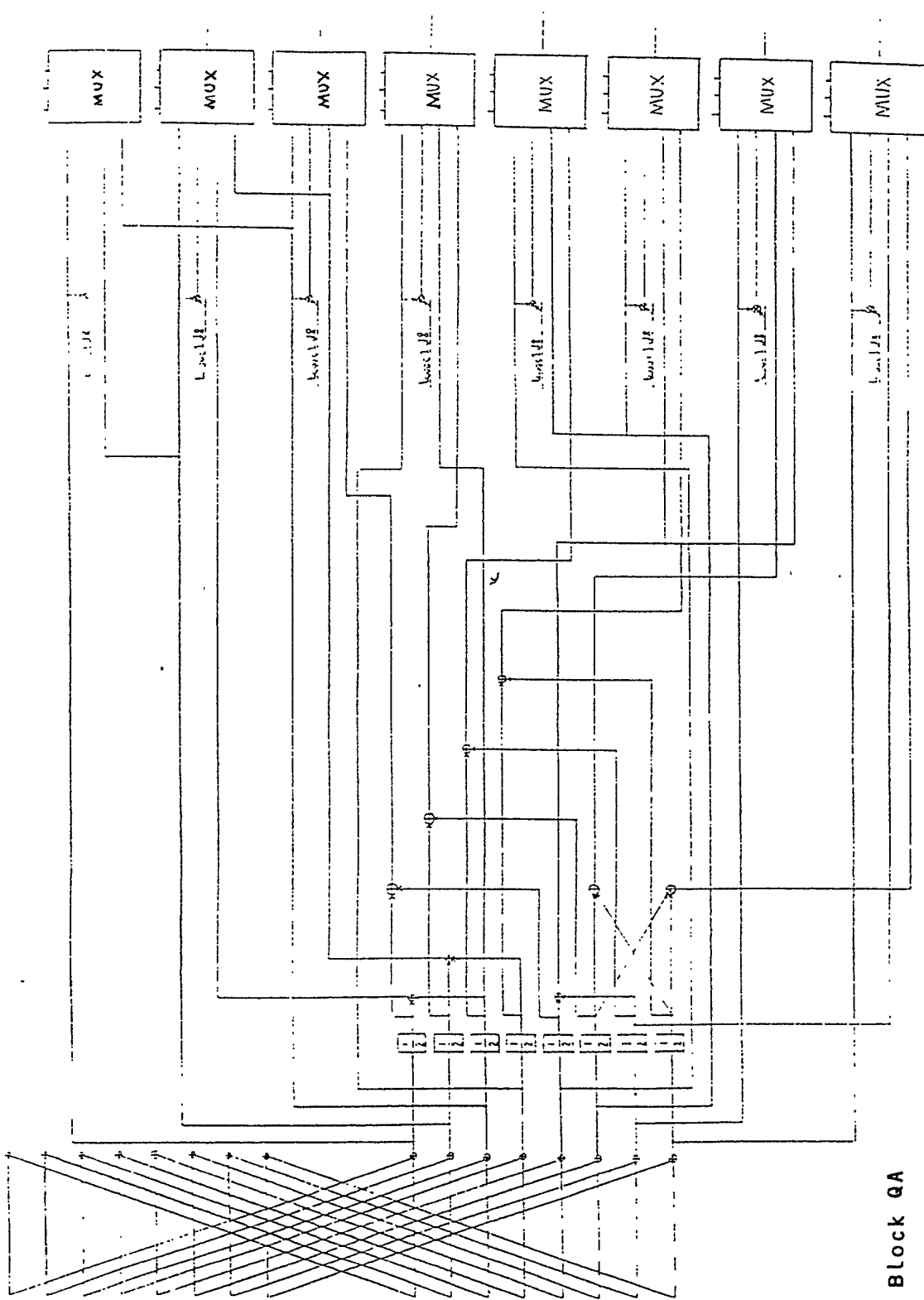


FIG. 16

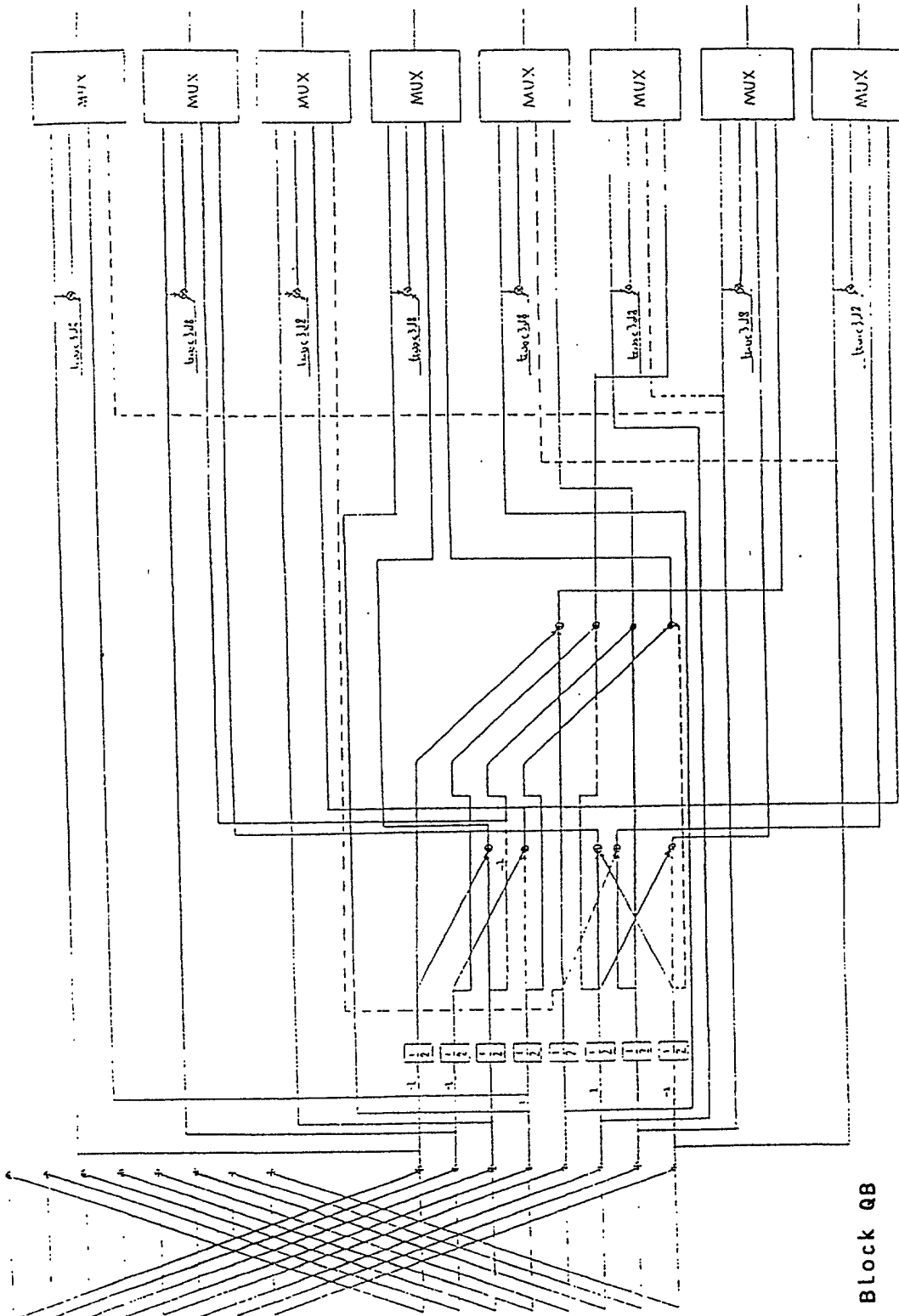


FIG. 17

66000-150660

18/24

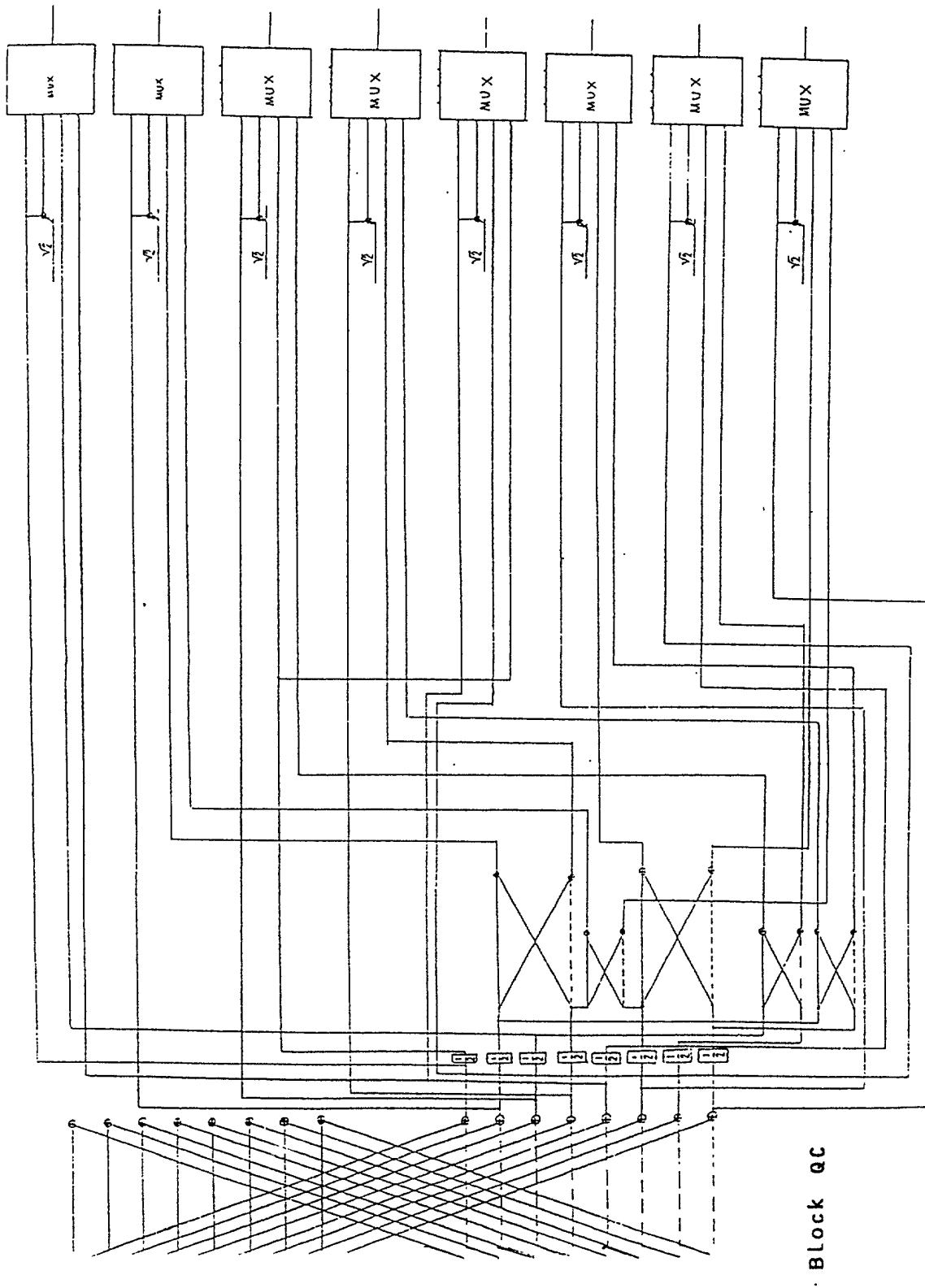


FIG. 18

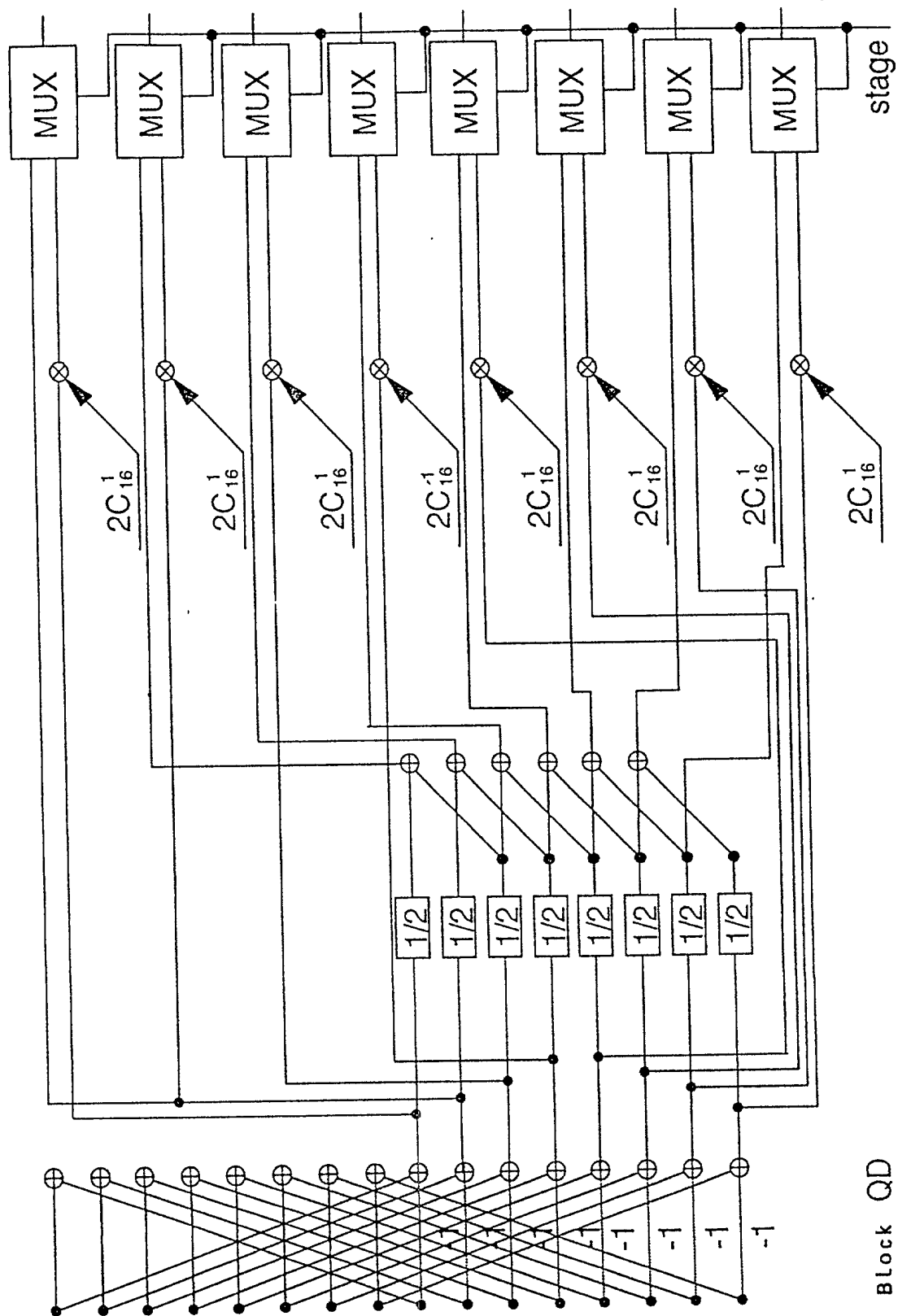
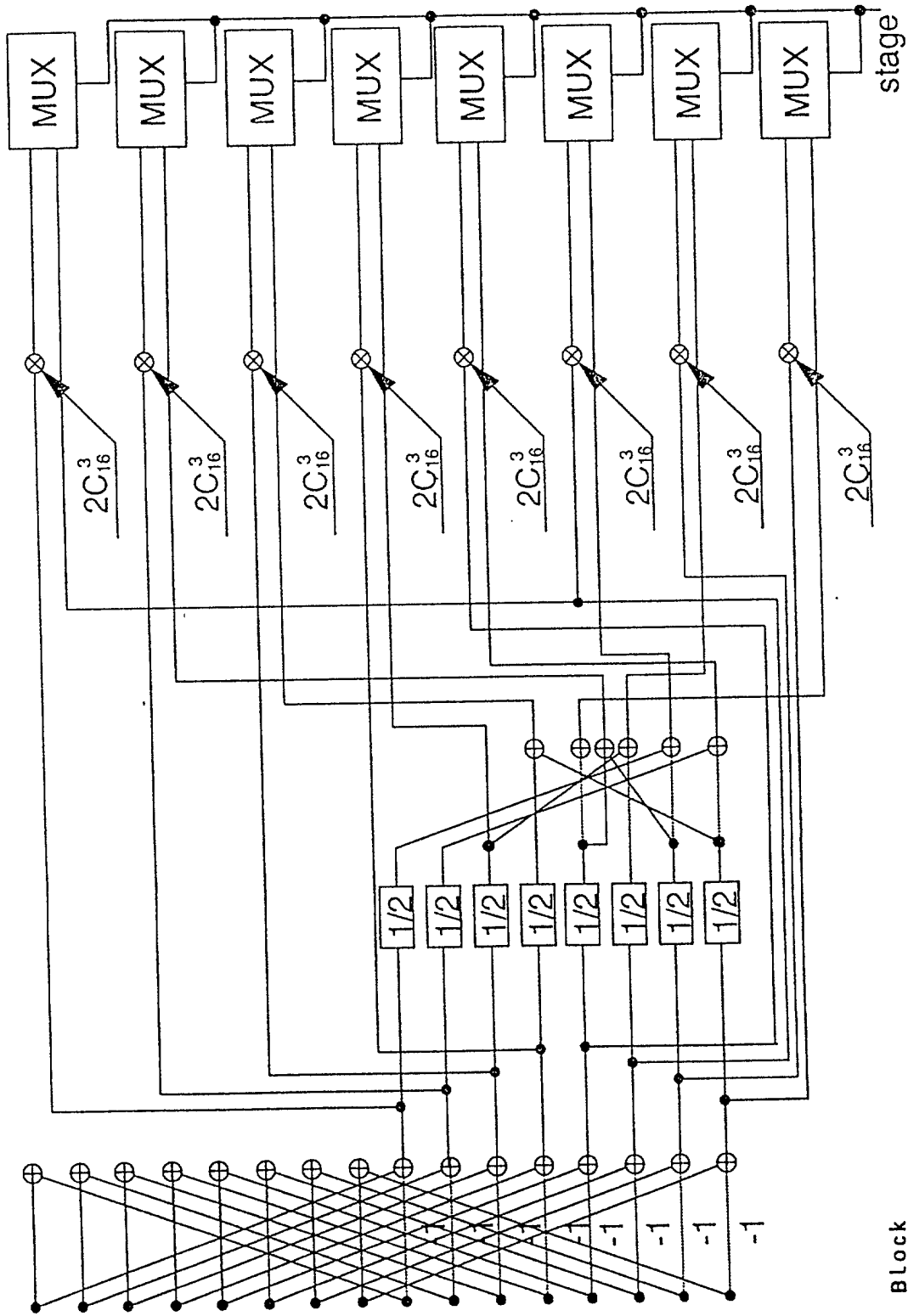


FIG. 19



Block
QE

FIG. 20

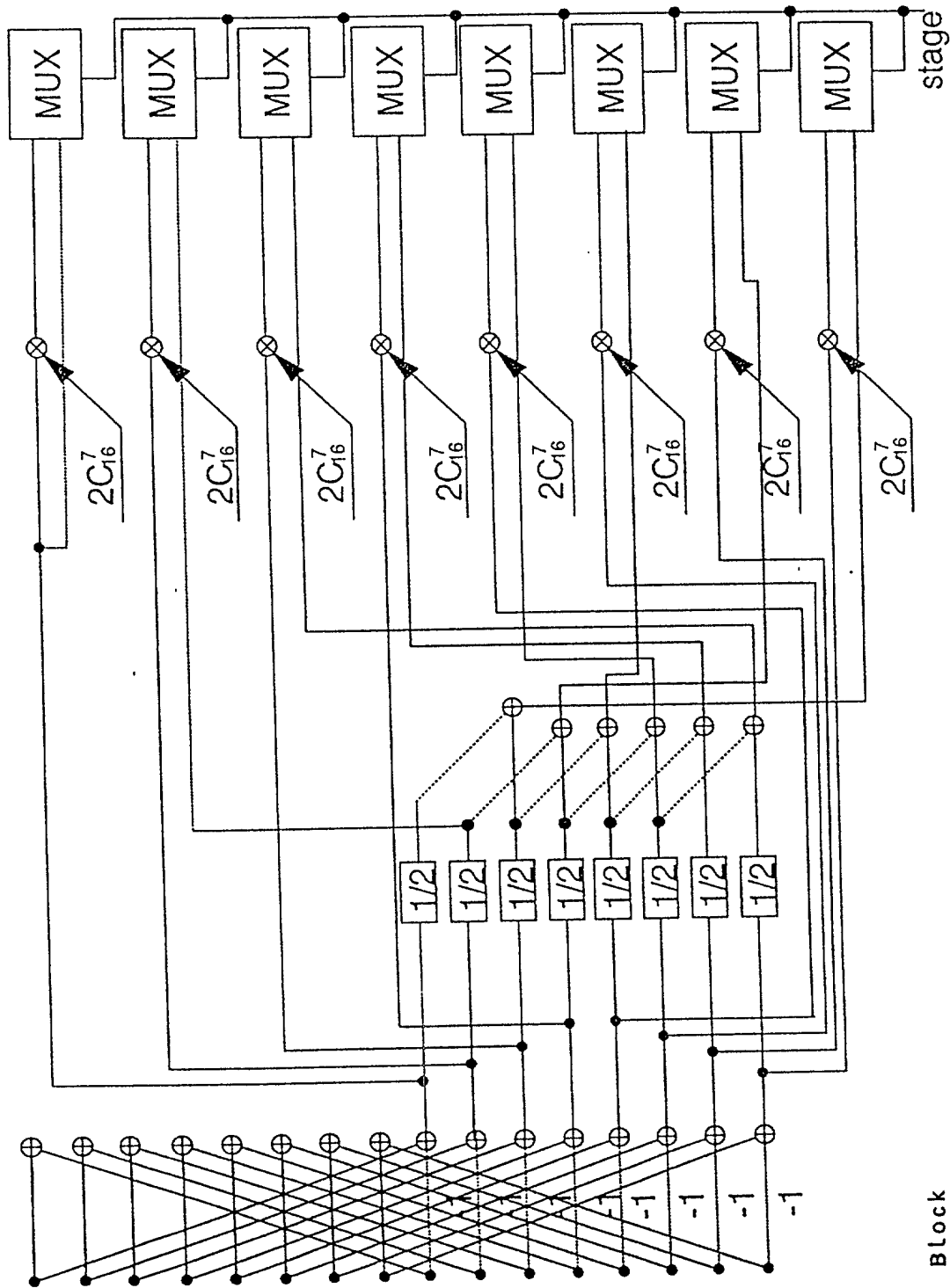


FIG. 21

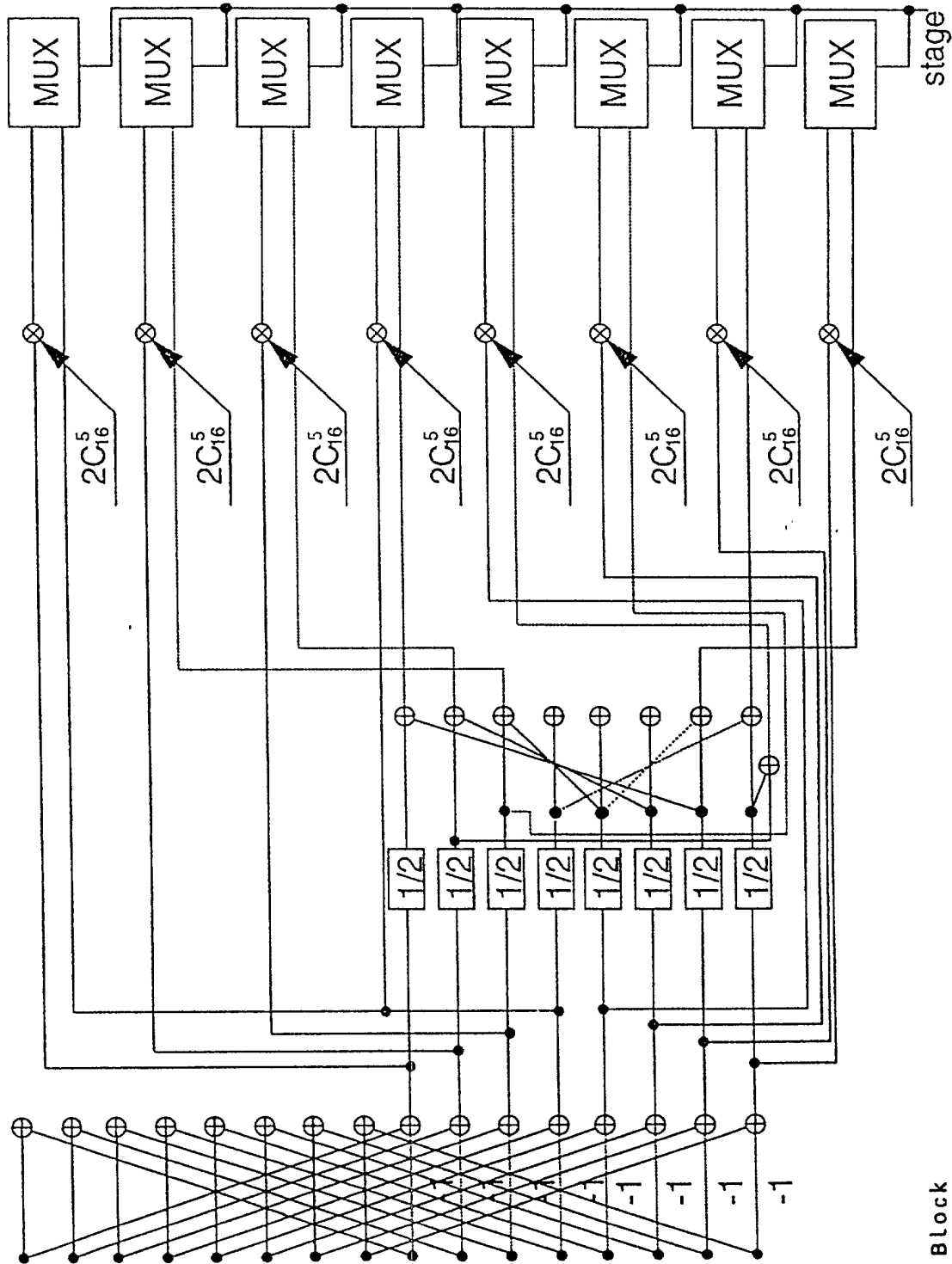


FIG. 22

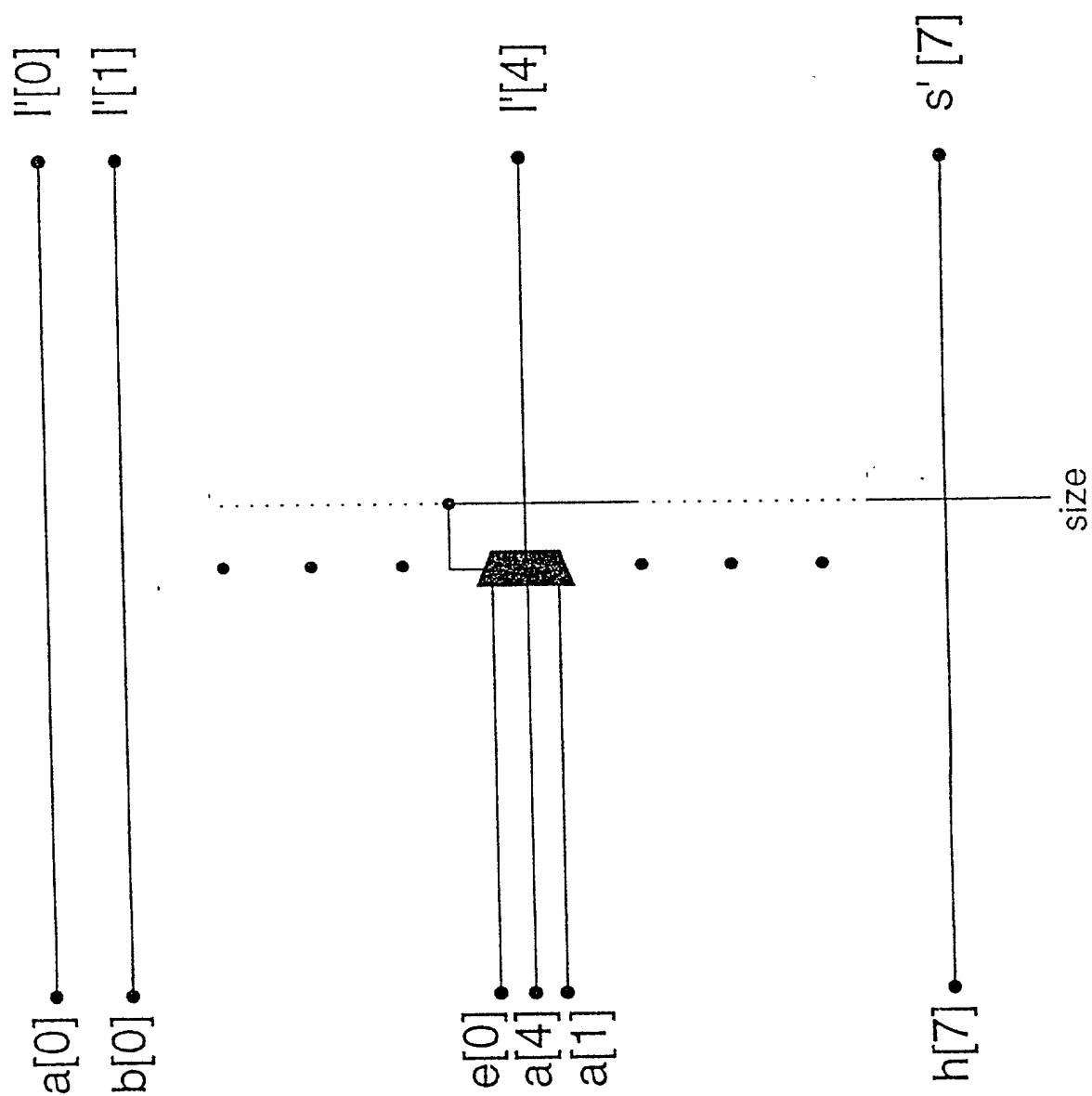
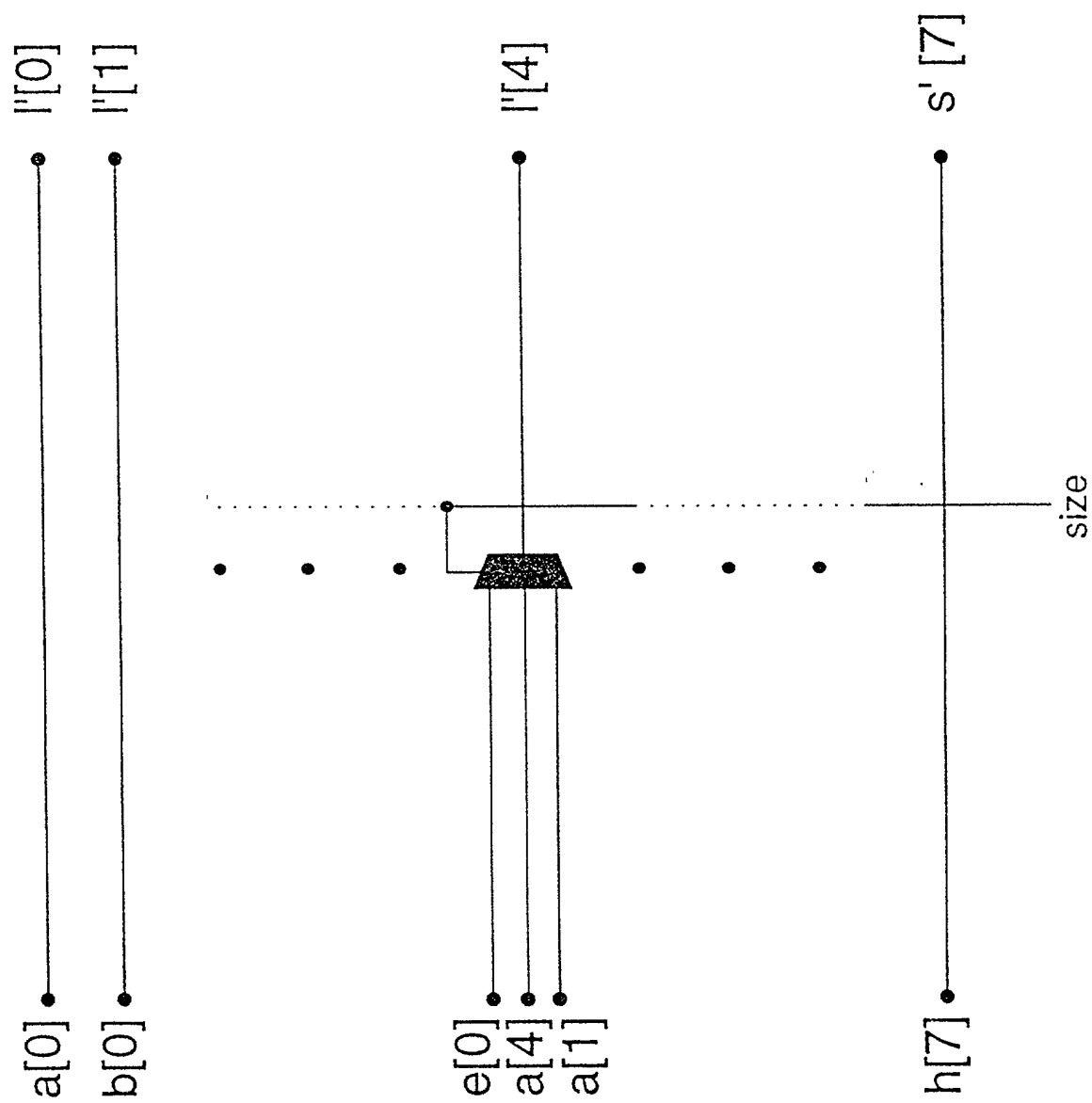


FIG. 23



~~FIG. 23~~

F16.23

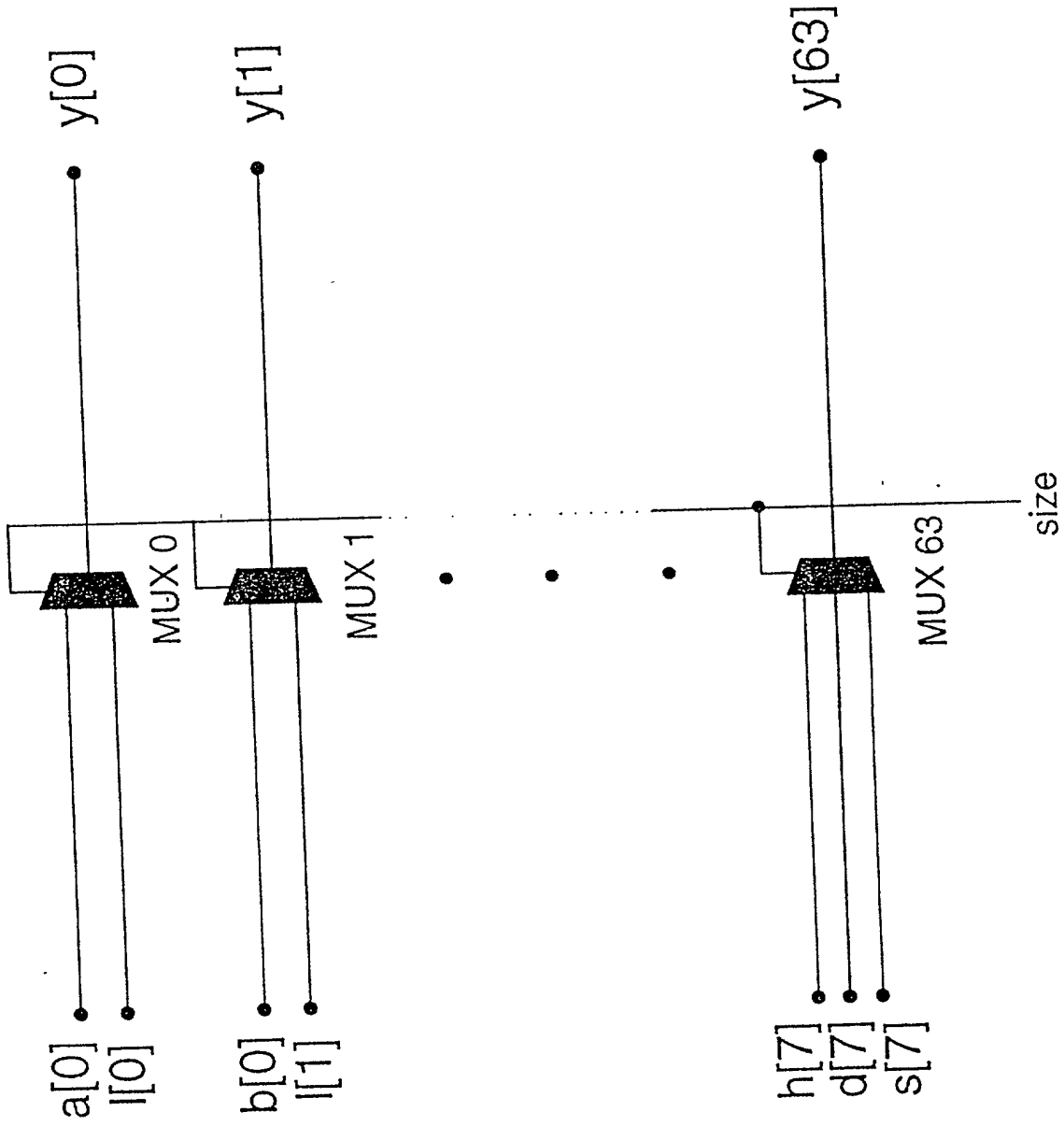
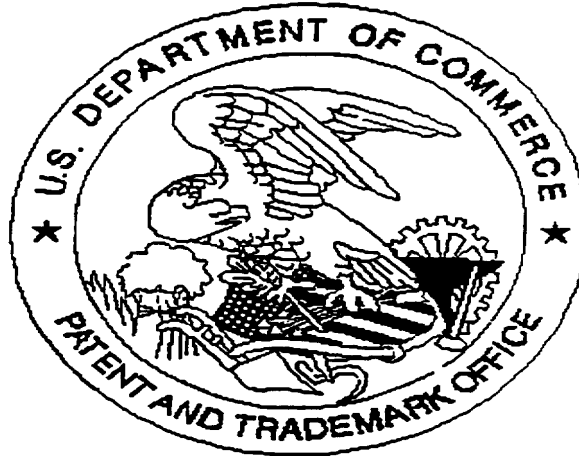


FIG. 24

United States Patent & Trademark Office
Office of Initial Patent Examination — Scanning Division



Application deficiencies were found during scanning:

☒ Page(s) 4 of Specification were not present
for scanning. (Document title)

☐ Page(s) _____ of _____ were not present
for scanning. (Document title)

☐ Scanned copy is best available.
